MICRO APPLICATION

APPLE 2
TRUCS ET ASTUCES
POUR APPLE 2E, 2+, 2C

UN LIVRE DATA BECKER

MICRO APPLICATION

APPLE 2
TRUCS ET ASTUCES
POUR APPLE 2E, 2+, 2C

UN LIVRE DATA BECKER

Distribué par MICRO APPLICATION
147 Av. Paul Doumer
92500 RUEIL MALMAISON

et également
EDITIONS RADIO
3 rue de l'Eperon
75006 PARIS

(c) Reproduction interdite sans l'autorisation de MICRO APPLICATION.

"Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de MICRO APPLICATION est illicite (loi du 11 mars 1957, alinéa 1er de l'article 40).

Cette représentation ou reproduction illicite, par quelques procédés que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

La loi du 11 mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à l'utilisation collective d'une part, et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration".

ISBN 2-86899-021-5

Copyright (c) 1984 DATA BECKER
Merowingerstr. 30
4000 DUSSELDORF
Allemagne de l'Ouest

Copyright (c) Traduction française 1985 MICRO APPLICATION
147 Av. Paul Doumer
92500 RUEIL MALMAISON

Traduction française et mise en pages assurées par Michel VALENTIN
Anne-Marie KEOKOSAL

Edité par Frédérique BEAUDONNET
Léo BRITAN
Philippe OLIVIER

TABLE DES MATIERES — Pages

Remarques préliminaires 1

Chapitre 1: Eléments fondamentaux de BASIC

1.1 Introduction 5
1.2 BASIC 6
1.3 Entrées-sorties d'informations simples 8
1.4 Premières commandes 13
1.5 Branchements 14
1.6 Boucles 17
1.7 Entrée de données 18
1.8 Utilisation des fonctions de chaînes 21
1.9 Utilisation des disquettes 22
1.10 Compléments 25
1.11 Un exemple pour conclure 27

Chapitre 2: Les périphériques

2.1 Remarques préliminaires 37
2.2 L'imprimante 37
2.2.1 Edition d'un programme 39
2.2.2 Sortie de données 40

2.2.3 Les graphiques 41
2.3 Le lecteur de disquettes 46
2.3.1 Remarques préliminaires 46
2.3.2 Trucs pour l'initialisation 49
2.3.3 Utilisation de disquettes interactives 50
2.3.4 Exemple: Evaluations statistiques 61

Chapitre 3: Accès mémoire et langage machine

3.1 Accès direct à la mémoire 74
3.2 L'assembleur 102
3.2.1 Bases 102
3.2.2 Combinaison BASIC-ASSEMBLEUR 115
3.3 Sous-programmes en langage machine 124
3.4 Exemples d'utilisation 130

Chapitre 4: Programmes graphiques

4.1 Remarques préliminaires 139
4.2 Graphique normal 141
4.2.1 Modes texte et graphique 141
4.2.2 Couleurs et effets graphiques 143
4.2.3 Instructions graphiques 145
4.2.4 Exemple 1: Ligne écran 147
4.2.5 Exemple 2: Grille 148
4.2.6 Exemple 3: Démonstration de couleur 149

4.2.7 Exemple 4: Histogramme 150
4.2.8 Exemple 5: Jeu de balle 153
4.3 Graphiques haute résolution 156
4.3.1 Préparation d'un graphique haute résolution 157
4.3.2 Couleurs 158
4.3.3 Les instructions graphiques 159
4.3.4 Exemple 1: Système de coordonnées 160
4.3.5 Exemple 2: Fonction linéaire 161
4.3.6 Exemple 3: Le cercle 164
4.3.7 Exemple 4: Graphique en trois dimensions 167
4.4 Le mélange de texte et de graphique 169
4.4.1 Dessiner un caractère 170
4.4.2 Programme d'aide pour le mélange de texte et de graphique 173

Chapitre 5: Amélioration des structures d'un programme

5.1 Remarques préliminaires 174
5.2 Les sous-programmes 175
5.3 Techniques de menu 188
5.4 Fusion de programmes 201

Chapitre 6: Edition

6.1 Remarques préliminaires 216
6.2 Le mode ESCAPE 217

6.3 Le programme LINE EDITOR 221

Chapitre 7: Aides à la programmation

7.1 Remarques préliminaires 237
7.2 Renumérotation et fusion de programmes 238
7.3 Vue d'ensemble et documentation 253
7.3.1 Contrôle des variables 254
7.3.2 Contrôle des adresses de branchement 261
7.3.3 Modifications 264
7.4 Initialisation 269
7.5 Présentation de l'écran 284
7.5.1 Positionnement sur l'écran 284
7.5.2 Fonctions de chaînes 287
7.5.3 Sorties d'écran formatées 290
7.5.4 Les masques d'écran 295
7.6 Classement 304

Chapitre 8: Alternative pour la programmation en BASIC

8.1 Remarques préliminaires 312
8.2 Autres langages de programmation 313
8.3 Les logiciels 333

Annexe I: Messages d'erreurs 337

Annexe II: Sous-programmes importants 341

Annexe III: Adresses mémoire 342

Annexe IV: Organisation de la mémoire de l'APPLE 346

Remarques préliminaires

Quiconque s'est déjà servi pendant un moment d'un petit ordinateur tel que l'APPLE II, se rend compte au bout d'un certain temps que les réponses données aux questions typiques des usagers dans les manuels ou les livres concernant les micro-ordinateurs sont trop limitées.

C'est pour cette raison, que dans ce livre, nous avons inclus quelques Trucs et Astuces qui faciliteront la programmation de l'APPLE II ou qui vous aideront à résoudre certains problèmes.

Il va de soi qu'en ce qui concerne le choix de ces problèmes nous avons dû opérer une certaine sélection. Il y a en effet de nombreux problèmes spécifiques, mais la place dont nous disposons nous oblige à une certaine sélection. Celle-ci a été basée sur une expérience de plusieurs années concernant les problèmes concrets que la mise en service de l'APPLE II a posée.

Il apparaît ainsi rapidement que lors de l'utilisation de l'ordinateur, les mêmes problèmes se reproduisent et c'est ceux-là que nous allons traiter.

Avant que le lecteur puisse se consacrer entièrement à ces problèmes quelques remarques préliminaires s'imposent si on désire utiliser les programmes présentés.

1. La toute première des remarques c'est que les programmes sont écrits dans la langue de l'ordinateur: le BASIC. Les autres langages ne seront pris en considération que dans quelques chapitres.

2. Nous partons du principe que le lecteur a déjà une expérience en BASIC. Néanmoins pour donner aux débutants la possibilité de suivre les programmes présentés ou de pouvoir les utiliser, nous allons consacrer un chapitre au rappel des instructions fondamentales de BASIC.

Le "connaisseur" peut tranquillement passer ce chapitre.

3.Pour les problèmes particuliers qui seront abordés ici, nous n'emploierons qu'exceptionnellement l'aide de logiciels commerciaux exception faite de certains utilitaires.
Notre but, est beaucoup plus de partager les " Trucs & astuces " avec les possesseurs d'APPLE qui n'accordent aucune valeur aux programmes du commerce (souvent chers) ou qui ne désirent pas les concevoir.

4.Lors de la sélection des thèmes que nous allons aborder ici, nous ne nous sommes pas laisser guider par les intérêts d'un cercle restreint d'usagers. Il ne s'agira pas d'un problème commercial précis, ni de statistiques ou d'un autre sujet spécifique.
Nous nous limiterons plus à la logique de la programmation, à son aspect méthodique et aux différentes difficultés que peuvent rencontrer les utilisateurs dans différents champs d'application.

5. En ce qui concerne les programmes Trucs et Astuces, ils n'utiliseront pas les possibilités les plus performantes et la vitesse de traitement ceci ne jouant pas pour nous un rôle prépondérant.
Il nous apparaît plus important que ceux qui élaborent un programme reconnaissent les différentes étapes.

6. Il est clair que nous ne détailleront pas toutes les finesses mais que les explications seront abordables pour le débutant (nous faisons confiance au "connaisseur" qui saura se servir des particularités et des spécificités

de l'APPLE II sans d'autres explications).

En résumé nous ne présenterons pas que des solutions mais nous proposerons à ceux qui progressent seuls, une méthode qui leur permettra de résoudre leurs problèmes, seuls.La résolution d'un problème ne réside pas dans l'établissement du programme mais beaucoup plus dans le fait de reconnaître la structure de ce problème, de l'analyser, de garder en tête la solution, de manière à ce qu'avant qu'un programme soit établi, cette solution lui soit transmise.

La poursuite de ce but impose des restrictions quant à l'énoncé de problèmes typiques si l'on veut permettre l'établissement d'une programmation structurée.

Cela signifie que les programmes décrits dans les chapitres à suivre respecteront les règles suivantes:

1. Présentation du problème sous différents aspects.
2. Développement des programmes adéquats.
3. Description des programmes et, le cas échéant, présentation des résultats.

Depuis quelques temps est apparu sur le marché un nouvel ordinateur de la famille APPLE II: l'APPLE IIC. Semblable à ses prédécesseurs, l'APPLE IIC profite d'une extension mémoire de 128 Ko.
En conséquence, il est à peu près certain que ,pour le lecteur de ce livre, l'APPLE IIC n'est pas un ordinateur entièrement nouveau.

Bien qu'il soit livré avec le système d'exploitation ProDOS, l'ancien DOS

3.3 reste encore disponible. Aussi tous les programmes développés ici sur la base du DOS 3.3 pourront être utilisés sur ce nouvel ordinateur même s'ils utilisent des instructions d'accès direct aux mémoires ou des routines en langage machine. De toutes manières les fichiers DOS 3.3 pourront être transformés en fichiers ProDOS grâce à un utilitaire; le nom du fichier sera transformé, en effet, ProDOS n'admet que 15 signes pour le nom. Les espaces et la ponctuation seront refusés. Lors de la conversion, les espaces seront transformés en points.
Il n'a pas toujours été possible de relever les détails du nouveau système d'exploitation, mais le BASIC APPLESOFT est de toutes façons resté le même. Une différence réside lors de la mise en oeuvre des deux lecteurs de disquettes. L'APPLE IIC possède un lecteur incorporé, un second en option et des lecteurs externes jusqu'à PR 7.
Ceci correspond à la numérotation usuelle des slots, même si le IIC ne bénéficie plus d'emplacements pour connecteurs.
Sous ProDOS, la commande INIT disparaît et un utilitaire permet le formatage (pour 3 systèmes d'exploitation différents). Un deuxième utilitaire permet la recopie de fichiers des deux types sur une nouvelle disquette.

En appliquant le chapitre 6 au IIC, le lecteur doit être prudent, notamment en ce qui concerne le programme LINE EDITOR.
Il en est de même pour le paragraphe "Initialisation" du chapitre 7 qui n'est pas intéressant pour les utilisateurs du système ProDOS.
Il est également possible sous ProDOS de démarrer un programme avec le "bootage" d'une disquette (c'est sensiblement la même chose que le programme d'initialisation).Il doit alors s'appeler STARTUP.

Pour finir nous vous rappelons que ce livre a été écrit par deux auteurs différents. De ce fait certaines différences d'ordre stylistique n'ont pu être entiérement évitées.

Chapitre 1:Eléments fondamentaux de BASIC

1.1 Introduction

Dans ce chapitre nous allons donner un aperçu des éléments de BASIC les plus importants. Nous considérons ce chapitre comme un rappel de connaissances déjà acquises. Ainsi, le débutant sera en mesure de comprendre les indications du chapitre suivant. Il se peut que pour lui ce rappel du BASIC soit un peu rapide pour; beaucoup de choses s'éclaireront dans les chapitres suivants avec l'établissement de programmes concrets.De sorte que ce qui paraissait obscur sera expérimenté grâce à des explications détaillées et à des exemples de programmes.

Comme dans les remarques préliminaires nous signalons que certains oublis ont été nécessaires: nous ne parlerons pas ici de toutes les programmations possibles mais seulement des plus importantes et des instructions les plus fréquentes.

Enfin certaines instructions seront exposées rapidement dans l'introduction et seront reprises dans les chapitres suivants si elles s'avèrent nécessaires à l'établissement d'un programme.

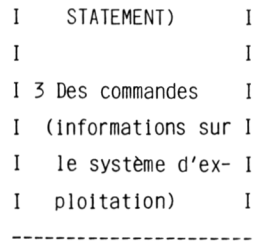
1.2 BASIC

Lorsqu'un ordinateur doit résoudre un problème, il a besoin pour cela d'une suite d'instructions appelée: programme.
Un tel programme doit être écrit dans une langue que l'ordinateur peut comprendre.
De nos jours la langue la plus usitée par les petits ordinateurs est le BASIC.

```
--------------------
I BASIC=Beginner's  I
I       All-Purpose I
I       Symbolic    I
I       Instruction I
I       Code        I
--------------------
```

Ce langage n'est pas standardisé mais il existe différents dialectes de BASIC. De toutes manières de nombreux éléments de base de ce langage se recoupent; l'APPLE II travaille avec son propre BASIC (BASIC APPLESOFT) qui recouvre certaines significations particulières que nous pourrons utiliser.
Avant de parler de ces particularités remarquons un fait important: toute sortie d'information rend nécessaire l'entrée de trois groupes d'informations:

```
----------------------
I 1 Des données      I
I                    I
I 2 Des instructions I
I   (appelées aussi  I
```



Les données peuvent être des chiffres, des nombres, des lettres ,des mots , des textes ou d'autres informations.

Les instructions engendrent une suite d'informations qui seront nécessaires à l'emploi des données désirées. En BASIC, ces instructions sont numérotées à la file avec des numéros de lignes (généralement par pas de 10).

Les commandes sont les points de rencontre entre le système d'exploitation d'un ordinateur et l'organisation des sorties de données du système informatique.
Les commandes n'utilisent pas de lignes numérotées.

1.3 Entrées-Sorties d'informations simples.

Pour la sortie d'informations, l'instruction d'affichage PRINT peut être de différentes façons.
Ceci est illustré par les exemples suivants:

```
10 PRINT 3            Impression du chiffre 3 à l'écran

20 PRINT 3+5          Impression du résultat 8

30 PRINT "APPLE"      Impression de la chaîne APPLE

40 PRINT X            Impression de la variable X

50 PRINT              Impression d'une ligne vide
```

Les possibilités décrites dans les instructions 10 à 40 peuvent être inclues ou mélangées dans une seule instruction PRINT. On doit alors les séparer par une virgule ou un point-virgule.

```
10 PRINT"SOMME=";3+5  Cette instruction donne d'abord la chaîne SOMME=
                      suivi du résultat 8;donc SOMME=8

20 PRINT 3,8          Impression des chiffres 3 puis 8 donc:
                      3     8
```

(Si l'on veut utiliser le nombre décimal 3.8 il faudra mettre un point à la place de la virgule.)

Si le point-virgule sépare les termes, les virgules séparent par contre les énoncés.

Chaque instruction PRINT provoque l'apparition d'une nouvelle ligne ,il faut donc que chaque instruction se termine par une virgule ou un point-virgule.

Le mot PRINT peut être abrégé par ?.

Si plusieurs instructions, comme par exemple l'instruction PRINT, doivent tenir sur une ligne ,on les sépare alors par :.

```
10 ?3+5              Impression de:8
20 ?
30 ?11-2                          9
```

Cette suite d'instructions peut être réunie, comme décrit plus haut, en une seule ligne:

```
10 ?3+5:?:?11-2
```

Si l'on souhaite des décalages dans l'impression, on utilisera la fonction TAB:

```
10 ?TAB(5)3          Impression du chiffre 3 à la colonne 5
```

Pour l'entrée des informations, on dispose des instructions INPUT et LET.

```
10 LET X=3.2         La variable X aura pour valeur 3.2
```

```
20 LET Y=2*X          La variable Y aura pour valeur 6.4

30 LET Y=Y+1          La variable Y aura maintenant la valeur 7.4

40 LET N$="TOTO"      La variable N$ aura pour valeur la chaîne TOTO
```

Le nom de la variable à gauche du signe = doit impérativement commencer par une lettre.

Le signe $ indique que la variable est une chaîne qui doit être entre guillemets lors de l'entrée des informations.

Le mot LET peut être supprimé dans une instruction LET:

```
10 XX=(3+5)/2         La variable XX a pour valeur 4

20 Y2=XX**2           La variable Y2 a pour valeur 16
```

(**=puissance)

Une série de fonctions peuvent être utilisées avec LET

```
10 LET X=SQR(16)      Calcule de la racine carrée de 16(=4);
                      c'est cette valeur qui va sortir.

20 ?X
```

Les autres fonctions importantes sont:

```
SIN, COS, TAN         :Fonctions trigonométriques
INT                   :Partie entière d'un nombre
```

```
ABS                    :Valeur absolue
LOG                    :Logarithme naturel
EXP                    :Fonction exponentielle
RND                    :Calcule un nombre aléatoire
```

Avec la commande LET, on dispose de la commande INPUT pour l'entrée des informations.

```
10 INPUT X             :Demande une valeur pour la variable X

20 INPUT A,B,C         :Demande trois valeurs

30 INPUT A$,B$         :Demande deux chaînes

40 INPUT"S.V.P. UN NOMBRE:";X
                       :Comme 10 mais avec affichage d'un texte

50 INPUT"S.V.P. VOS NOM ET PRENOM:";N$,P$
                       :Comme 30 mais avec affichage d'un texte
```

On fera attention avec l'instruction INPUT de donner à l'ordinateur autant de valeurs (numériques ou chaînes selon les variables) que de variables; de manière à ce qu'à chaque variable soit attribuée une valeur. Les noms des variables sont à séparer d'une virgule de même que les valeurs qui leur ont été attribuées.

Pour que le sens d'un programme reste entier il faut le clore avec l'instruction END.

Les commentaires, remarques, titres ou équivalents peuvent être ajoutés grâce à l'instruction REM.

Exemple:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT :PRINT TAB(9)"R.PRUST/W.WOSS 1984"
50 PRINT : PRINT TAB(11)"1.1 - PYTHAGORE"
60 PRINT : PRINT : PRINT : PRINT : PRINT
70 INPUT "PREMIER COTE DE L'ANGLE DROIT:";A
80 INPUT "DEUXIEME COTE DE L'ANGLE DROIT:";B
90 C=SQR(A**2+B**2)
100 PRINT : PRINT : PRINT"HYPOTHENUSE  =";C
110 PRINT : PRINT "FIN"
120 END
```

1.4 Premières commandes

Un programme BASIC comme il a été déjà expliqué, est d'abord traduit par l'ordinateur quand nous envoyons au système d'exploitation la commande:

RUN

Lorsqu'on désire voir un programme après son lancement et sa disparition de l'écran, on a besoin de la commande:

LIST

La commande LIST peut se rapporter à une ligne ou à un groupe de lignes, comme le montrent les exemples suivants:

LIST 50-150	:Les lignes 50 à 150 seront listées
LIST -80	:Toutes les lignes jusqu'à la ligne 80 seront listées
LIST 100-	:A partir de la ligne 100 toutes les lignes seront listées
LIST 60	:La ligne 60 sera listée

Si le contenu de la mémoire doit être changé, par exemple pour l'établissement d'un nouveau programme, il faut utiliser la commande:

NEW

1.5 Branchements

Il arrive fréquemment que certaines parties de programme doivent être lues plusieurs fois. Pour que ce soit possible, on a besoin d'instructions de branchement de deux types: les branchements conditionnels et inconditionnels.

Un branchement inconditionnel s'obtient grâce à l'instruction GOTO comme le montre l'exemple suivant:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT TAB(10)"1.2 - BOUCLE SANS FIN"
60 PRINT : PRINT : PRINT : PRINT : PRINT
70 I=1
80 Q=I*I
90 PRINT I,Q
100 I=I+1
110 GOTO 80
120 END
```

Ce programme imprime tous les entiers naturels et leur carré aussi longtemps que la capacité mémoire de l'ordinateur le permet (donc très longtemps!).

On peut arrêter le déroulement d'un programme (comme aussi celui des listings) en appuyant sur

CTRL et C

Si l'on veut reprendre le programme à l'endroit où il a été interrompu il faut utiliser la commande

CONT

Pour ralentir le programme ou le listing on peut utiliser

CTRL et S

Les branchements conditionnels (instruction IF...THEN), se différencient des branchements inconditionnels parce qu'ils ne s'exécutent que si certaines conditions préalables sont remplies.

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT : PRINT TAB(9)"1.3 - BRANCHEMENT CONDITIONNEL"
60 PRINT : PRINT
70 I=1
80 Q=I*I: PRINT I,Q : I=I+1
90 IF I =12 THEN GOTO 80
100 PRINT"FIN" : END
```

Dans ce programme les entiers naturels et leur carré ne seront donnés qu'entre I=1 et I=12. Puis le programme se termine.

Ceci est valable dans tous les cas: si la condition logique après le mot clé IF est remplie, l'ordinateur exécutera les instructions qui suivent le THEN. Il peut y avoir ici aussi plusieurs instructions qui se suivent, elles seront alors séparées par ":".

Si par contre la condition n'est pas remplie, le programme se poursuit à la

ligne suivante.

Dans quelques cas il peut être nécessaire de ménager certaines parties de programme (nommées sous-programmes ou subroutines) pour y sauter en cas de besoin.

C'est l'instruction de branchement GOSUB qui permet ceci. Le retour au programme principal s'obtient par l'instruction RETURN, le retour se fait à l'instruction qui suit immédiatement le GOSUB.

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT : PRINT TAB(10)"1.4 SOUS-PROGRAMMES"
60 PRINT : PRINT : PRINT : PRINT
70 GOSUB 1000 : REM ENTREE
80 I=1 : GOSUB 2000
110 REM ETC.
998 PRINT : PRINT : PRINT"FIN"
999 END
1000 REM SP ENTREE
1010 PRINT : PRINT : PRINT
1020 INPUT"CHOISISSEZ TROIS NOMBRES:";X,Y,Z
1030 RETURN
2000 REM SP CALCUL
2010 S=X+Y+Z
2020 AM=S/3
2025 PRINT : PRINT : PRINT
2030 PRINT I;" VALEUR MOYENNE :";AM
2040 RETURN
```

1.6 Boucles

Il est possible de répéter une partie de programme avec les instructions FOR...TO et NEXT comme l'illustre l'exemple suivant:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT : PRINT TAB(12)"1.5 - BOUCLE"
60 PRINT : PRINT
70 FOR I=1 TO 12
80 PRINT I,I*I
90 NEXT I
100 PRINT : PRINT"FIN": END
```

Ici encore on affichera les entiers naturels de 1 à 12 et leur carré.
Il est possible de fixer un pas de progression différent de 1:

```
    10 FOR I=1 TO 10 STEP 0.4
```

1.7 Entrée de données

L'entrée de données par les instructions LET ou INPUT est compliquée pour les données permanentes. L'instruction DATA combinée à l'instruction READ est mieux adaptée surtout si on emploie l'instruction DIM en plus.

L'instruction DIM permet de réserver un emplacement mémoire pour le (ou les noms) nom de variables:

```
10 DIM X(20),Y(15)          :Emplacement réservé pour X 21,pour Y 16
```

Les données sont préparées par l'instruction DATA puis elles sont lues par l'instruction READ. Voir l'exemple suivant:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT : PRINT TAB(13)"1.6 - DONNEES"
60 PRINT : PRINT
70 DIM X(15)
80 DATA 88,66,77,75,63,69,84,87,72,79,75,78,74,91,83
90 FORI=1 TO 15 : READ X(I) : NEXT I
100 PRINT"CONTROLE:" : PRINT
110 FOR I=1 TO 15: PRINT X(I);" ";: NEXT I: PRINT : PRINT
120 REM SUITE DU TRAITEMENT DES DONNEES
130 PRINT : PRINT : PRINT"MAINTENANT LES DONNEES PEUVENT ETRE TRAITEES"
140 PRINT : PRINT"L'ACCES VA ETRE FERME."
150 PRINT : PRINT : PRINT"FIN" : END
```

Dans ce programme X(I) est appelé variable indicée; les seules valeurs lues sont disponibles sous le nom X(0), X(1)...

Les variables peuvent être indexées de plusieurs façons. Ainsi, dans

l'exemple suivant, on utilise une double indexation:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT TAB(12)"1.7 - EMBOITEMENT"
60 PRINT : PRINT
70 DATA 172,68
80 DATA 183,85
90 DATA 180,81
100 DIM X(3,2)
110 FOR I=1 TO 3
120 FOR J=1 TO 2
130 READ X(I,J)
140 NEXT J
150 NEXT I
160 PRINT : PRINT"CONTROLE:": PRINT
170 FOR I=1 TO 3
180 FOR J=1 TO 2 : PRINT X(I,J);" "; : NEXTJ : PRINT : PRINT
190 NEXT I
200 PRINT : PRINT"LE TRAVAIL SUR LES DONNEES PEUT ETRE EFFECTUE"
210 PRINT : PRINT"FIN": END
```

Dans cet exemple, on a réservé 4*3=12 emplacements de mémoire. Les variables s'appellent X(0,0), X(0,1), X(0,2), X(1,0)...
Le premier index s'appelle ligne d'indexation, le second colonne d'indexation.

On remarque qu'après READ les données ne peuvent plus être lues.
Il est possible de poursuivre la lecture de données dans une liste de DATA à l'endroit où la dernière donnée a été lue.
Si les données doivent être relues depuis le début avec READ, on utilisera l'instruction RESTORE.

1.8 Utilisation des fonctions de chaînes

Une chaîne est une suite de signes qui contient des lettres, des chiffres ou des signes particuliers. Ces chaînes sont employées, en BASIC, entre guillemets.

Avec l'aide des opérations de comparaisons logiques (, ,=, =, =), les chaînes peuvent être comparées dans leur suite alphabétique les unes aux autres.
Avec le signe + , les chaînes peuvent être concaténées.
Les fonctions suivantes permettent de travailler sur les chaînes. Ces fonctions sont expliquées ici rapidement et les exemples sont succins, mais nous trouverons suffisamment d'illustrations dans les chapitres suivants.

LEN : Cette fonction donne la longueur d'une chaîne, donc le nombre de signes la composant.
Exemple: L=LEN("BONJOUR") attribue à la variable L la valeur 7.

VAL : Cette fonction convertit une chaîne comportant des chiffres en valeur numérique.
Exemple: V=VAL("1789 REVOLUTION") attribue à la variable V la valeur 1789.

STR$: Cette fonction convertit une valeur numérique en chaîne.
Exemple: A$=STR$(1789) attribue à la variable A$ la chaîne "1789".

LEFT$: Cette fonction donne les x premiers caractères d'une chaîne.

Exemple: B$=LEFT$("PAQUES",3) attribue à B$ la chaîne "PAQ".

RIGHT$: Cette fonction donne les y derniers caractères d'une chaîne.
Exemple: C$=RIGHT$("W.VOSS",4) attribue à la variable C$ la chaîne "VOSS".

MID$: Cette fonction utilise trois facteurs:
1. La chaîne à traiter
2. Le numéro du caractère à partir duquel le découpage va avoir lieu.
3. La longueur du tronçon à découper.

Exemple: D$=MID$("BONJOUR",3,2) attribue à la variable D$ la chaîne "NJ".

ASC : Cette fonction renvoie le code ASCII du caractère dans ASC.
Exemple: X=ASC("A") attribue la valeur 65 à X.

CHR$: Cette fonction convertit le code ASCII en caractère.
Exemple: A$=CHR$(66) attribue à A$ le caractère "B".

1.9 Utilisation des disquettes

Les disquettes permettent de mémoriser en permanence des données et des programmes.

Pour "fabriquer" de nouvelles disquettes, il faut d'abord les initialiser (ou les formater) et suivre les règles suivantes:

1. Introduire le système d'exploitation de la disquette dans le lecteur (disquette DOS.MASTER)

2. Mise en marche de l'ordinateur

3. Insérer une nouvelle disquette

4. Entrer par exemple le programme suivant:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT TAB(9)"1.8 - INITIALISATION"
60 PRINT : PRINT
70 REM ************************
80 REM CONTENU DU PROGRAMME D'INITIALISATION
90 PRINT"CONTENU DU PROGRAMME D'INITIALISATION:" : PRINT : PRINT
100 PRINT"DISQUETTE NO 1"
110 PRINT"VOSS, MAI 1984"
120 END
```

5.Mise en route de la commande

INIT Nom

à la place de"Nom", un nom de programme doit être attribué tel que P1 ou équivalent.

Remarque: Chaque programme mémorisé doit avoir un nom. C'est ce que nous appelons un fichier.

```
-----------------------------------------------------------------
I  ATTENTION Lorsque la lampe rouge du lecteur s'allume il ne faut I
I            en aucun cas ouvrir la porte !                      I
-----------------------------------------------------------------
```

On peut maintenant transférer des programmes en mémoire sur la disquette initialisée.

Exemple:

SAVE PROGR1

Si on désire lire le programme, on utilise:

LOAD PROGR1

Si on veut avoir la liste des fichiers mémorisés sur la disquette, il suffit de faire:

CATALOG

Si on désire détruire un fichier, on utilisera la commande

DELETE Nom de fichier

Il arrive parfois que certains fichiers doivent être rebaptisés, on utilisera alors:

RENAME ancien Nom,nouveau Nom

1.10 Compléments

Lorsqu'un programme donne de nombreux résultats, il arrive que ceux-ci disparaissent vers le haut de l'écran. On peut y remédier en stoppant le déroulement du programme grâce à l'instruction

STOP

Un programme suspendu par l'instruction STOP peut être remis en route par la commande CONT.

Exemple:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT : PRINT TAB(15)"1.9  - STOP"
60 PRINT : PRINT
70 FOR I=1 TO 35
80 Q=I*I : PRINT I,Q
90 IF I/10=INT(I/10) THEN PRINT "TAPEZ CONT": STOP
100 NEXT I
110 PRINT"FIN": END
```

Dans des cas semblables, on peut utiliser l'instruction GET. Pour poursuivre son travail, l'ordinateur attend la frappe d'une touche.

Exemple:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT : PRINT TAB(13)"1.10 ATTENDRE"
60 PRINT : PRINT : PRINT"J'ATTENDS.": PRINT : PRINT "APPUYEZ SUR UNE TOUCHE
!" : PRINT : PRINT : PRINT
70 GET A$ : IF A$="" THEN 70
80 PRINT : PRINT : PRINT"LA TOUCHE ";A$;" A ETE APPUYEE."
90 PRINT : PRINT"MAINTENANT, ON PEUT CONTINUER."
100 REM SUITE DU PROGRAMME
110 PRINT : PRINT : PRINT"FIN": END
```

L'instruction HOME doit être mentionnée, elle a pour effet de renvoyer le curseur au coin gauche de l'écran (position HOME) et d'effacer l'écran. Cette instruction est importante pour la présentation.

Des résultats équivalents peuvent être obtenus avec les instructions VTAB et HTAB.

VTAB Z indique que l'impression pour la prochaine instruction PRINT se fera à la colonne Z, HTAB S indique que l'impression se fera à la ligne S.

1.11 Un exemple pour conclure

Ce chapitre va être clos avec un exemple important illustrant les éléments fondamentaux de BASIC.

Définition du problème

On se propose de développer un programme qui trie par ordre croissant un ensemble de données quelconques.

Analyse du problème

L'ordinateur ne peut résoudre que des problèmes que nous avons déjà presque résolus dans notre tête. Ce qui signifie que la solution doit nous apparaître clairement avant d'écrire la programmation d'un problème à résoudre.

C'est cette pré-structuration du problème qu'on appelle analyse. Elle est particulièrement détaillée en ce qui concerne la question du processus de classement, c'est-à-dire le noyau du programme à développer.
Cette analyse peut être conduite avec des variables quelconques.
Classer signifie que nous allons comparer deux données entre elles. Nous allons comparer la première valeur avec toutes les autres. Si la première valeur est supérieure à celle que l'on compare, il faudra les inverser toutes les deux. Ainsi lorsque la première valeur aura été comparée à toutes les autres (après le premier tour), la plus petite valeur se trouvera en première position.
Ensuite, on compare la valeur en deuxième position avec toutes les autres,

à l'exception de la première. Si la deuxième valeur est supérieure à celle que l'on compare, on les inverse .
On obtiendra ainsi à la fin du deuxième tour que la valeur la plus petite soit en deuxième position.
Ensuite, on compare la valeur en troisième position avec toutes les autres, sauf avec les deux précédentes. A la fin de ce troisième tour on a trouvé et placé en troisième position la plus petite valeur.

Il peut y avoir autant de tours que nécessaire.

Si nous avons ainsi quatre valeurs, trois tours sont nécessaires. Au premier tour trois comparaisons ont lieu, au deuxième, deux comparaisons et au troisième, il reste une valeur à comparer.

Voici schématiquement le déroulement:

```
Tour 1 :  Comparaison 1 :  Valeur 1 avec Valeur 2
          Comparaison 2 :  Valeur 1 avec Valeur 3
          Comparaison 3 :  Valeur 1 avec Valeur 4

Tour 2 :  Comparaison 1 :  Valeur 2 avec Valeur 3
          Comparaison 2 :  Valeur 2 avec Valeur 4

Tour 3 :  Comparaison 1 :  Valeur 3 avec Valeur 4
```

Déroulement du programme

Le mieux pour expliquer le déroulement est de s'attribuer des données arbitraires.

Par exemple, on désire trier les cinq données suivantes:

6 3 5 9 7

On désignera sous le nom Champ la position d'une valeur.

Tour 1. (I=1)	Avant	Après
Comparaison 1 (J=2)		
(Champ 1 avec Champ 2)	6 3 5 9 7	3 6 5 9 7
Comparaison 2 (J=3)		
(Champ 1 avec Champ 3)	3 6 5 9 7	3 6 5 9 7
Comparaison 3 (J=4)		
(Champ 1 avec Champ 4)	3 6 5 9 7	3 6 5 9 7
Comparaison 4 (J=5)		
(Champ 1 avec Champ 5)	3 6 5 9 7	3 6 5 9 7
		Impression X(1)=3
Tour 2. (I=2)		
Comparaison 1 (J=3)		
(Champ 2 avec Champ 3)	3 6 5 9 7	2 5 6 9 7
Comparaison 2 (J=4)		
(Champ 2 avec Champ 4)	3 5 6 9 7	3 5 6 9 7
Comparaison 3 (J=5)		
(Champ 2 avec Champ 5)	3 5 6 9 7	3 5 6 9 7
		Impression X(2)=5
Tour 3. (I=3)		
Comparaison 1 (J=4)		

(Champ 3 avec Champ 4) 3 5 6 9 7 3 5 6 9 7

Comparaison 2 (J=5)

(Champ 3 avec Champ 5) 3 5 6 9 7 3 5 6 9 7

Impression X(3)=6

Tour 4. (I=4)

Dernier Tour

Comparaison 1 (J=5)

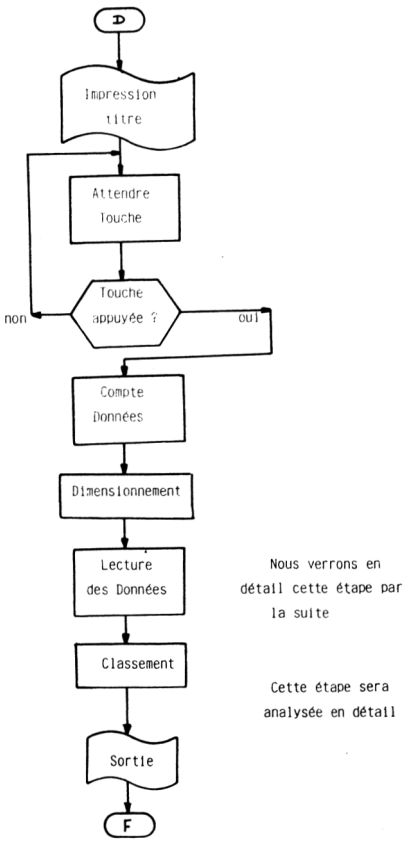
(Champ 4 avec Champ 5) 3 5 6 9 7 3 5 6 7 9

Impression X(4)=7

Fin de la boucle I;

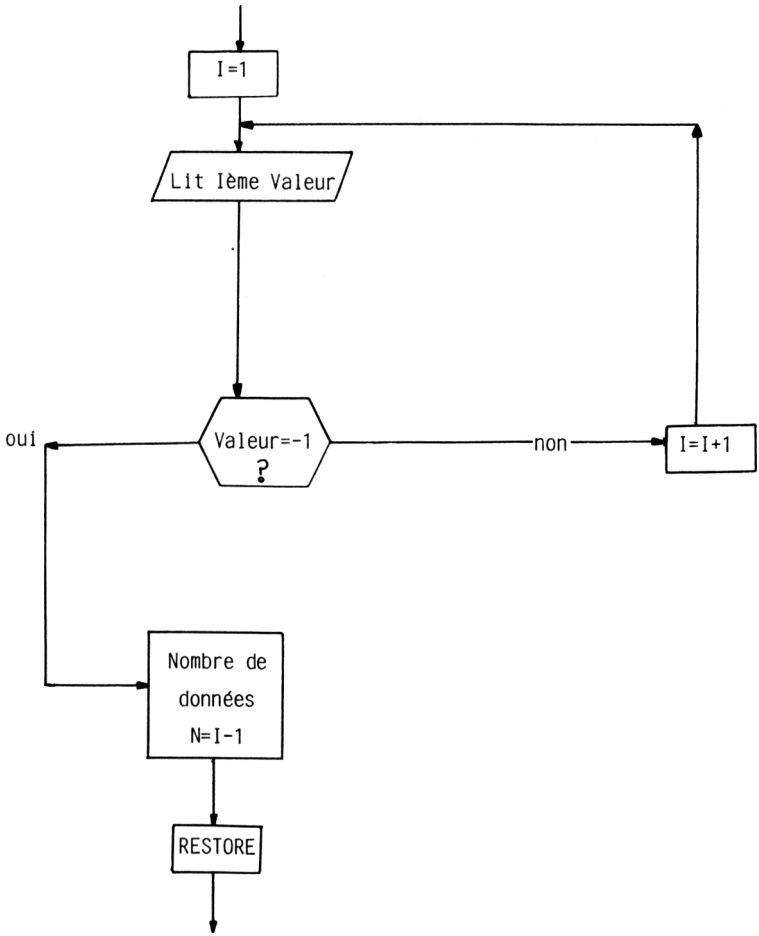
Organigramme

Le déroulement du programme se présente très clairement avec un organigramme (plan de déroulement d'un programme) et les étapes particulières d'un programme sont apparentes. Le développement d'un tel organigramme est recommandé pour l'établissement d'autres programmes.
Dans ce cas l'organigramme suit les règles suivantes:



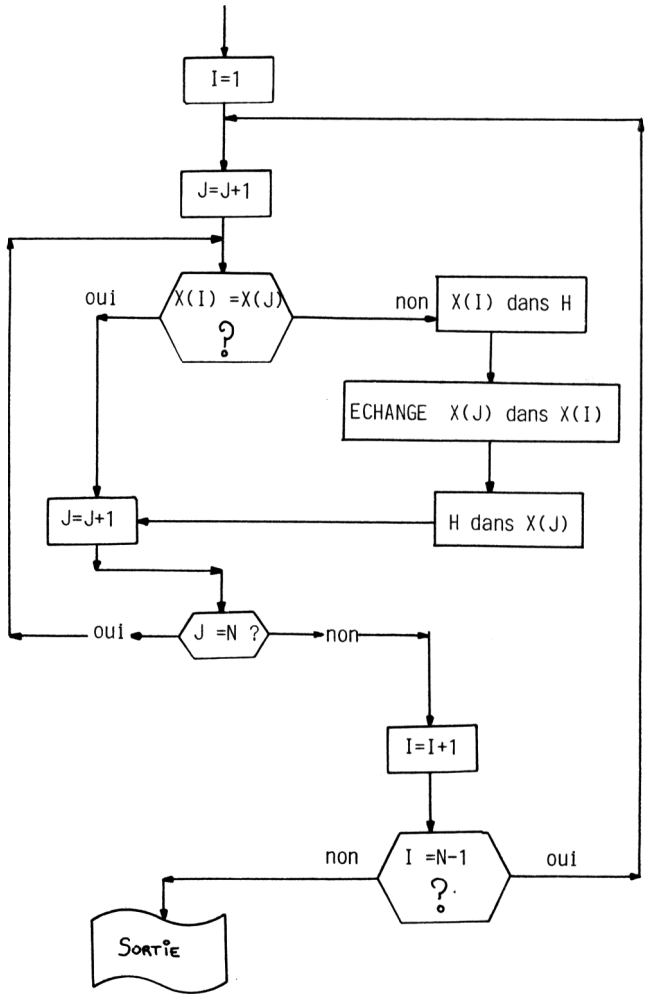
Comptage des données

-1 indique la fin des données.



Classement

N=Nombre de valeurs



Programme

```
30 PRINT TAB(9)"APPLE TRUCS ET ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT TAB(11)"1.11 - CLASSEMENT"
60 PRINT : PRINT : PRINT : PRINT
70 PRINT "LE PROGRAMME SUIVANT CLASSE"
80 PRINT "UN ENSEMBLE DE DONNEES QUELCONQUES."
90 PRINT : PRINT
100 PRINT "LES DONNEES STOCKEES DANS LES DATAS"
110 PRINT "(INSTRUCTIONS 500...)" : PRINT
120 PRINT "LA DERNIERE VALEUR DOIT ETRE ABSURDE"
130 PRINT" ICI: -1."
140 PRINT : PRINT : PRINT TAB(6)"APPUYEZ SUR UNE TOUCHE S.V.P."
150 GET A$:IF A$="" THEN 150
160 HOME
170 REM COMPTAGE DES DONNEES
180 FOR I=1 TO 9999 : READ Z : IF Z=-1 THEN 200
190 NEXT I
200 N=I-1
210 PRINT "NOMBRE DE VALEURS= ";N
220 REM LECTURE DES DONNEES
230 RESTORE
240 DIM X(N)
250 FOR I=1 TO N : READ X(I) : NEXT I
260 REM IMPRESSION DE CONTROLE
270 PRINT : PRINT : PRINT" CONTROLE: " : PRINT
280 FOR I=1 TO N : PRINT X(I);" "; : NEXT I : PRINT
```

```
290 REM CLASSEMENT ET IMPRESSION"
295 PRINT : PRINT : PRINT
300 PRINT : PRINT"CLASSE: " : PRINT
310 FOR I=1 TO N-1 : FOR J=I+1 TO N
320 IF X(I) X(J) THEN 340
330 H=X(I) : X(I)=X(J) : X(J)=H
340 NEXT J : PRINT X(I);" ";
350 NEXT I : PRINT X(I)
360 PRINT : PRINT : PRINT" FIN DE L'IMPRESSION" : END
500 REM DONNEES
510 DATA 7,3,2,9,8,1,4
520 DATA -1
530 END
```

Cet exemple montre très clairement comment un problème relativement simple et vu dans son ensemble peut être traité par un ordinateur.

Nous vous rappelons que nous n'avons pas cherché à élaborer un programme particulièrement rapide et élégant. Ca ne sera pas le cas non plus dans les chapitres suivants.

Chapitre 2: Les périphériques

Celui qui achète un petit ordinateur se contente souvent en premier lieu (et pas seulement pour des raisons financières!) de l'unité centrale elle-même, d'un écran comme unité de sortie et du clavier qui est inclus pour l'APPLE.

On s'apercevra rapidement que l'on ne peut se passer d'une imprimante comme organe de sortie supplémentaire et d'au moins un lecteur de disquettes comme élément de mémoire périphérique.

Avec ces périphériques s'ouvrent de nombreuses possibilités dont nous allons parler ici.

2.2 L'imprimante

Avec une imprimante, il est possible de lire "noir sur blanc" les listings ou les résultats d'un programme.

Dans le système APPLE on doit ajouter à l'imprimante une carte de contrôle dans le Slot 1 (Faire attention aux conseils d'installation!).

D'autres cartes de connexion seront nécessaires pour d'autres éléments périphériques. On appelle INTERFACES ces cartes de connexion.

Lors du transfert des informations de l'ordinateur vers les périphériques (ou l'inverse), on utilisera la commande

PRINT Numéro

en mode direct, avec le numéro du connecteur.

En mode programme la sélection des connecteurs est un peu plus compliquée. La commande PR 1 (SLOT 1) doit être employée dans une instruction PRINT dans laquelle la combinaison des touches

CTRL et D

sera donnée dans un symbôle, ce symbôle n'apparaît pas à l'écran.

Voici un programme typique:

```
10 D$=""   Entre guillemets
           le signe correspondant
           à l'appui des touches
           CTRL et D
20 ?D$;"PR 1"

...
```

Maintenant toutes les sorties PRINT se font sur l'imprimante.

Il existe deux façons d'utiliser une imprimante:
soit en mode commande pour l'édition d'un programme
soit en mode programme pour l'édition de données d'un programme.

2.2.1 Edition d'un programme

Avec les commandes adaptées il est aisé de lister un programme sur imprimante au lieu de l'écran:

```
PR 1
LIST
PR 0
```

Il n'y a rien d'autre à ajouter à ceci.

2.2.2 Sortie de données

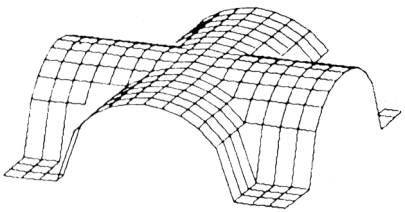
Nous avons déjà vu que la combinaison des touches CTRL et D dans un PRINT correspondait à la commande PR 1.

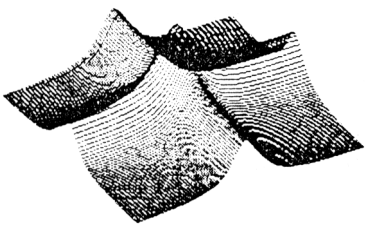
Voici un exemple pour illustrer ceci:

```
30 PRINT TAB(9)"APPLE TRUCS ET ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST/W.VOSS 1984"
50 PRINT : PRINT TAB(11)"2.1  IMPRIMANTE"
60 PRINT: PRINT: PRINT: PRINT
70 D$="": REM CTRL ET D
80 PRINT D$;"PR 1"
90 PRINT "CE TEXTE SORTIRA"
100 PRINT"SUR IMPRIMANTE."
110PRINT D$;"PR 0"
120 END
```

2.2.3 Les graphiques

Les graphiques établis par l'Apple doivent pouvoir être obtenus, en cas de besoin, sur l'imprimante. Ce n'est pas un problème avec les imprimantes matricielles actuelles car chaque point à l'écran peut être représenté. Ainsi on peut former ce genre d'images :





La programmation de telles éditions est longue et complexe.
Il n'y a cependant pas de problème particulier à l'établissement de graphiques sur écran (cf. Chapitre 4), en effet une série d'instructions facilitent la programmation et font gagner un temps précieux.

Il en va autrement de la sortie de ces graphiques sur imprimante. Sur le papier, les positions particulières de l'impression doivent être établies de façon très précise . Une telle impression graphique n'est en fin de compte rien d'autre qu'une accumulation de points noirs dans un certain ordre. Pour travailler avec des adresses hexadécimales codifiées on doit avoir en tête les classifications de mémoire. C'est pourquoi il est recommandé de mettre en place un programme d'aide ou un matériel de traitement complémentaire qui permette d'établir ces sorties organisées.

L'Apple possède une "Carte Interface Imprimante Parallèle" qui a des propriétés remarquables: cette carte permet aussi bien la sortie de textes que la sortie graphique à haute définition (voir chapitre 4) .
Cette carte s'insère dans le connecteur 1 et doit être reliée à l'imprimante (lire attentivement les conditions d'utilisation). Les commandes mises en oeuvre par cette carte sont envoyées grâce à la touche CTRL.
Si on est en mode programme et non en mode direct, on utilisera la fonction CHR$. A la commande CTRL-I correspond l'instruction:

```
PRINT CHR$ (9).
```

Les instructions les plus courantes sont les suivantes :

```
PR 1        : Mise en route de l'interface
PR 0        : Arrêt de l'interface
CTRL-I80N   : Longueur des lignes de l'imprimante = 80 signes
```

(Elles pourraient être d'une longueur différente).
Avec 00 à la place de 80: annulation de la fonction.
Ces instructions peuvent être groupées dans une ligne
de programme de la manière suivante :
10 PRINT CHR$ (9);"80N"

CTRL-I I : Tous les signes seront affichés sur l'écran et sur l'imprimante.

CTRL-I5L : Saut de 5 lignes

CTRL-I70R : A partir de la colonne 70 il faut faire attention aux espaces. Si on en trouve il se produit un interlignage (on peut changer la valeur de 70).

CTRL-160P : Fixe 60 lignes par page (emplacement alternatif) suivent alors automatiquement 6 interlignages (hors service avec 00)

CTRL-IS : Impression du texte à l'écran.

Voici quelques applications simples à titre d'exemple :

```
80 PR  1
90 PRINT CHR$ (9) ; "50N"
95 PRINT CHR$ (9) ; "20L""
97 PRINT CHR$ (9) ; "I"
100 PRINT "CE PROGRAMME MONTRE DE QUELLE  MANIERE  LA  CARTE  INTERFACE  DE
L'IMPRIMANTE"
110 PRINT "PEUT, COMME ON L'A EXPLIQUE PLUS HAUT ETRE UTILISEE."
120 PRINT : PRINT : PRINT : PRINT "FIN": END
130 PR  0
```

Résultat:

```
CE PROGRAMME MONTRE DE QUELLE
MANIERE LA CARTE IMPRIMANTE
```

```
PEUT, COMME ON L'A EXPLIQUE PL
US HAUT, ETRE UTILISEE.



FIN
```

Si le lecteur à la fin de ce programme donne l'ordre CTRL-I S . Qu'arrive-t-il ?

Les instructions les plus importantes pour la sortie graphique sont les suivantes :

Le contenu de la mémoire graphique peut chez Apple, comme nous l'avons déjà expliqué, être directement utilisé avec cette interface. Pour cela nous avons exclusivement besoin de la commande:

CTRL-I et "G" plus Argument

ou dans un programme:

```
(nn) ?CHR$(9);"G Argument"
```

L'argument précise une des options suivantes (La liste n'est pas complète, seules les possibilités les plus intéressantes sont précisées).

2 : Impression de la deuxième page écran (Auparavant : première page)
D : Sortie du graphique avec une hauteur double
I : Inversion du graphique (Auparavant les points blancs à l'écran étaient imprimés en noir).
R : Rotation du graphique de 90 deg.
A : Les deux pages de l'écran sont reliées (AND)

(Pour des explications supplémentaires se reporter au chapitre 4)

Avec ces instructions un RESET est nécessaire au cas où, pour un travail précis il n'y aurait pas assez de place sur l'imprimante. Nous reviendrons dans le chapitre 4 sur les possibilités graphiques de l'APPLE.

2.3. Le lecteur de disquettes

2.3.1 Remarques préliminaires

L'essentiel des mémoires externes pour les systèmes de micro-ordinateur sont les disquettes qui peuvent être lues et écrites par un lecteur de disquettes approprié.

A l'inverse de nombreux autres systèmes, les lecteurs de disquettes APPLE n'ont pas leur propre prise de courant mais sont connectés sur l'alimentation de l'ordinateur. Cela implique que lorsque l'ordinateur est allumé, le lecteur se met aussi en marche. Ainsi si on a inséré une disquette initialisée les parties nécessaires pour l'exploitation d'une disquette sont emmagasinées par le système d'exploitation de manière à ce qu'on puisse travailler tout de suite avec la (ou les) disquette. Cette manière de procéder est appelée démarrage à froid du système.

Si on a inséré une disquette non initialisée, on peut arrêter le lecteur en marche avec la touche RESET, avant d'avoir commencé à travailler avec l'ordinateur.
Le lecteur de disquettes peut être utilisé pour établir deux sorties différentes:
1) Pour la mémorisation et/ou la relecture d'un programme .
2) Pour l'utilisation interactive de fichiers de données avec une utilisation du lecteur pendant le déroulement du programme.

Nous allons nous occuper tout d'abord, avant de donner un exemple détaillé, d'aspects particuliers rencontrés lors du maniement des disquettes.

Il est important pour l'utilisation des disquettes de distinguer deux manières de mémoriser les informations :

1 La mémorisation séquentielle
2 La mémorisation aléatoire

Les explications de ces notions appliquées à la notion de fichiers nous donnent l'occasion d'esquisser quelques notions de base du travail des données :

On désire établir un questionnaire dans lequel les informations suivantes seront demandées:

1. Sexe
2. Date de naissance
3. Situation de famille

...etc...

Les données établies sont présentées sous la forme de matrices de données :

Nombre de pers.	Sexe	Date de naissance	Situation de famille	...
001	M	1942	marié	...
002	F	1953	célibataire	...
003	M	1963	célibataire	...
004	M	1958	marié	...
...				

Tous les autres procédés de travail sur les données peuvent être établis sur cette matrice de données. Ils sont identiques si les renseignements spécifiques sont employés comme plus haut ou lorsqu'ils sont transformés en valeurs numériques (exemple "masculin=0, féminin =1).

Chaque ligne de cette matrice s'appelle jeu de données, record ou set.

Chaque colonne s'appelle array. La zone de rencontre entre une colonne et une ligne s'appelle champ (Field).

Dans un champ on trouve une variable pour un porteur d'indices (exemple une personne).

Un champ comprend plusieurs symbôles (au moins un) et pour chacun de ces symbôles un octet est réservé dans l'ordinateur . L'ensemble de toutes les lignes (ou colonnes) de la matrice de données est appelée fichier (file)

Lors de la mémorisation séquentielle à l'intérieur des fichiers, les jeux de données succèdent à des jeux de données, par exemple des adresses de clients, à des adresses de clients: une adresse longue prend beaucoup de place, une courte moins.

Schématiquement c'est à peu près comme cela (D1,D2,D3.....sont des jeux de données différents)

```
-------------------------------------------------------------------
///// D1 //////// D2 //////// D3 /////// D4 /////// D5 //////////
-------------------------------------------------------------------
```

Dans un fichier de type séquentiel, la recherche d'un jeu précis de données demande du temps (par exemple l'adresse de Monsieur Dupont) car lors de la recherche il faut reprendre le fichier à son point de départ.

Lors de la mémorisation à accès direct (on parle aussi de mémoire à accès sélectif ou aléatoire) chaque jeu de données a la même longueur.
Schématiquement c'est à peu près ainsi:

```
-------------------------------------------------------------------
/// D1 /// D2 /// D3 /// D4 ///  ..............................
-------------------------------------------------------------------
```

On utilise alors plus de mémoire (même les adresses courtes demandent beaucoup de place) mais l'ordinateur a maintenant l'accès direct à des données précises, sans devoir pour cela lire les précédentes. Pour trouver un jeu de données précis (lors de l'utilisation de la mémoire d'une disquette) quelques fractions de secondes suffisent.

2.3.2 Trucs pour l'initialisation

On a déjà parlé dans le chapitre 1 des instructions, du lecteur de disquettes, les plus importantes. La commande INIT joue un rôle prédominant. Il faut formater une nouvelle disquette avant son utilisation.

Cette initialisation peut être effectuée de telle manière qu'à chaque mise en marche du lecteur avec cette disquette une impression communique des informations intéressantes... Il faudra alors utiliser l'instruction PRINT dans le programme d'initialisation.

2.3.3 Utilisation de disquettes interactives

Un des avantages de l'utilisation d'un ordinateur pour le traitement des données réside dans l'utilisation interactive de la mémoire externe.

Nous entendons par là, l'accès au fichier d'une disquette lors du déroulement d'un programme . Le programme peut utiliser des données stockées sur la disquette ou les résultats établis dans un programme peuvent être écrits pour archivage ou pour une utilisation ultérieure.

On a besoin pour l'utilisation d'une disquette de deux instructions BASIC supplémentaires OPEN et CLOSE. Ces deux instructions servent aussi à régulariser l'échange entre l'ordinateur et la mémoire externe, l'instruction OPEN ouvre le canal vers la disquette, il est refermé avec l'instruction CLOSE. Plus exactement on obtient avec OPEN et CLOSE l'ouverture et la fermeture d'un fichier donc d'une zone de la disquette qui englobe les données qui vont ensemble et auxquelles l'utilisateur devra donner un nom. Le nom d'un tel fichier (qui à l'inverse de beaucoup d'autres ordinateurs n'est pas entre guillemets) peut être long de 30 signes.L'exemple suivant montre comment un tel fichier peut être écrit (nous retournons à l'exemple de la matrice de données qui a été donnée plus haut).

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT:PRINT:PRINT TAB(9)"R.PRUST / W.VOSS 1984"
50 PRINT:PRINT TAB(6)"2.2 ECRIRE SUR UNE DISQUETTE"
60 PRINT:PRINT:PRINT:PRINT
80 PRINT"CE PROGRAMME MONTRE COMMENT DES DONNEES"
90 PRINT"(LIGNES DATA 500...) PEUVENT"
```

```
100PRINT"ETRE ECRITES SUR DISQUETTE"
120 GOSUB 1000:REM ATTENDRE
130 REM LECTURE
140 N=4
150 DIM NR(N),G$(N),JG(N),F$(N)
160 FOR I=1 TO N:READ NR(I),G$(I)),JG(I),F$(I)):NEXT I
170 HOME:PRINT"CONTROLE:":PRINT:PRINT
180 FOR I=1 TO N:PRINT NR(I);TAB(5)G$(I);TAB(15)JG(I));TAB(22)F$(I): NEXT I
190 PRINT:PRINT:PRINT"L'ECRITURE SE FAIT..."
200 D$=CHR$(4):REM CTRL ET D
210 PRINT D$;"OPEN FICHIER1"
220 PRINT D$;"WRITE FICHIER1"
230 FOR I=1 TO N:PRINT NR(I),G$(I),JG(I),F$(I):NEXT I
240 PRINT D$;"CLOSE FICHIER1"
250 PRINT:PRINT:PRINT"FIN DU TRANSFERT DES DONNEES"
260 END
500 DATA 001,MASCULIN,1942,MARIE
510 DATA 002,FEMININ,1953,CELIBATAIRE
520 DATA 003,MASCULIN,1962,CELIBATAIRE
530 DATA 004,MASCULIN,1958,MARIE
1000 REM SP ATTENDRE
1010 VTAB 23:HTAB 6:INVERSE:PRINT"APPUYEZ SUR UNE TOUCHE!"
1020 NORMAL
1030 GET A$:IF A$="" THEN 1030
1040 RETURN
```

Voici quelques explications:

Lignes	Contenu
30 30-60	Titre
80-100	Commentaire
120	Branchement dans SP 1000 (programme d'attente)
140-160	Lecture des données (information du nombre de jeu de données + boucle de lecture jusqu'à 4 variables)
170-180	Impression de contrôle des données
190	Notification du transfert sur la disquette
200	Introduction de la chaîne D$(l'impression des touches CTRL et D ensembles correspondent au code ASCII 4; c'est pourquoi on utilise simplement CHR$(4)
210	Dans la chaîne suivant D$ on ouvre le fichier dans lequel les données doivent être stockées.
220	De la même manière il faut écrire dans le fichier et pas seulement le lire.
230	Boucle d'écriture (toutes les instructions PRINT vont maintenant sur la disquette).
240	Fermeture du fichier (voir 210)
250-260	Fin de l'impression et fin
500-530	Données
1000	Titre du programme d'attente
1010	Information aux utilisateurs en inverse, ligne 23 colonne 6
1020	Retour au mode normal
1030	Attente de l'appui d'une touche
1040	Retour

De la même façon voici un programme permettant de lire un fichier:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT:PRINT:PRINT TAB(9)"R.PRUST / W.VOSS 1984"
50  PRINT:PRINT TAB(6)"2.3 LECTURE DE LA DISQUETTE"
60  PRINT:PRINT:PRINT:PRINT
70  DIM N$(4)
80  PRINT"CE PROGRAMME MONTRE COMMENT ON"
90  PRINT"PEUT LIRE DES DONNEES SUR UNE DISQUETTE"
110 DATA LEON,LOUIS,EMILE,RENE
120 FOR I=1 TO 4:READ N$(I):NEXT I
130 D$=CHR$(4)
140 PRINT D$;"OPEN FICHIER2"
150 PRINT D$;"WRITE FICHIER2"
160 FOR I=1 TO 4:PRINT N$(I):NEXT I
170 PRINT D$;"CLOSE FICHIER2"
190 REM LECTURE DE LA DISQUETTE
210 PRINT D$;"OPEN FICHIER2"
220 PRINT D$;"READ FICHIER2"
230 FOR I=1 TO 4:INPUT A$: N$(I)=A$: NEXT I
240 PRINT D$;"CLOSE FICHIER2"
250 HOME:PRINT"CONTROLE":":PRINT:PRINT
260 FOR I=1 TO 4
270 PRINT N$ (I)
280 NEXT I
290 PRINT:PRINT:PRINT"FIN DE L'EDITION":END
```

Description du programme

Lignes	Contenu
30-60	Titre
70	Dimensionnement
80-90	Commentaire
100-170	Création d'un fichier de textes (pour les particularités voir l'exemple précédent)
190-240	Lecture de la disquette (les instructions INPUT sont nécessaires pour cela)
250-280	Edition de contrôle
290	Fin du programme

Remarque :
Lors de l'exploitation de disquettes interactives, l'instruction GET conduit souvent à des erreurs. C'est pourquoi nous avons renoncé à un sous-programme d'attente, alors que nous en avions utilisé un dans l'exemple précédent.

On utilise donc lors de l'écriture d'une disquette en exploitation interactive, l'instruction WRITE et l'instruction INPUT lors de la lecture. Ces deux instructions apparaissent dans les PRINT comme les composantes des chaînes. Le lecteur attentif aura sûrement remarqué qu'il s'agissait de deux exemples d'utilisation de fichiers séquentiels.

Si l'on désire contrôler les échanges d'information entre l'ordinateur et le lecteur à l'écran, il est parfois intéressant d'employer l'instruction

MON C,I,O.

L'argument C indique que toutes les instructions du système d'exploitation en cours apparaissent sur l'écran rapidement ; I montre toutes les informations en entrée (Input); O montre toutes les opérations en sortie (Output). Certains de ces arguments peuvent être supprimés.

On arrête ces impressions avec l'instruction

NOMON

ou, en redémarrant la disquette. La commande NOMON peut comprendre les mêmes arguments que la commande MON.
Si elle contient moins d'arguments, seuls les effets respectifs seront annulés.

Si le lecteur utilise, pour essayer ce programme , une disquette sur laquelle d'autres programmes sont déjà mémorisés (grâce à la commande SAVE) il peut établir avec CATALOG, le relevé du contenu de la disquette qui indique d'un A un programme normal. Les autres fichiers qui ont été créés et établis sur une disquette pendant le déroulement du programme en exploitation interactive seront marqués dans le CATALOG d'un T.

On appelle de tels fichiers, "fichiers de textes". Ces fichiers peuvent être mémorisés avec LOAD, mais seulement dans le cadre du déroulement d'un programme, comme cela a été montré dans le dernier exemple.

Les fichiers de textes ainsi établis peuvent être facilement modifiés dans différentes directions:

1. Destruction d'un fichier.
2. AJoût de données dans un fichier.

3. Destruction de données dans un fichier.
4. Modification de données.

Les instructions nécessaires seront expliquées rapidement dans le prochain exemple.
La destruction de fichiers établis interactivement se réalise avec la fonction

DELETE nom de fichier.

L'exemple suivant montre comment on peut ajouter des données:

```
30  PRINT TAB (9) "APPLE TRUCS & ASTUCES"
40  PRINT:PRINT:PRINT TAB (9)"R.PRUST / W. VOSS 1984"
50  PRINT:PRINT:PRINT TAB (8)"2.4 AJOUT DE DONNEES"
60  PRINT:PRINT:PRINT:PRINT
70  DIM N$(4),B$(6)
80  PRINT "CE PROGRAMME MONTRE COMMENT AJOUTER"
90  PRINT "DES DONNEES SUR UNE DISQUETTE"
105 REM ECRIRE
110 DATA LEON,LOUIS,RENE,REMI
120 FOR I= 1 TO 4:READ N$(I):NEXT I
130 D$=CHR$ (4)
140 PRINT D$;"OPEN FICHIER3"
150 PRINT D$;"WRITE FICHIER3"
160 FOR J=1 TO 4:PRINT N$(I) : NEXT I
180 PRINT D$;"CLOSE FICHIER3"
190 REM LECTURE DE LA DISQUETTE
210 PRINT D$;"OPEN FICHIER3"
220 PRINT D$;"READ FICHIER3"
230 FOR I=1 TO 4
```

```
240 INPUT A$:B$(I)=A$
250 NEXT I
260 PRINT D$;"CLOSE FICHIER3"
270 REM IMPRESSION DE CONTROLE
280 HOME :PRINT "CONTROLE :": PRINT:PRINT
290 FOR I=1 TO 4
300 PRINT B$(I)
310 NEXT I
320 REM AJOUT
330 PRINT D$; "APPEND FICHIER3"
340 PRINT D$; "WRITE FICHIER3"
350 PRINT"ADAM": PRINT"EVA"
360 PRINT D$;"CLOSE FICHIER3"
370 PRINT:PRINT:PRINT"CONTROLE APRES AJOUT": PRINT
380 PRINT D$;"OPEN FICHIER3"
390 PRINT D$;"READ FICHIER3"
400 FOR I=1 TO 6: INPUT A$: B$(I)=A$: NEXT I
420 PRINT D$;"CLOSE FICHIER3"
430 FOR I=1 TO 6 : PRINT B$(I) : NEXT I
440 END
```

Voici ce qui se passe :

Lignes	Contenu
30-100	Titre, commentaire et dimensionnement
105-120	Préparation des données
130-180	Etablissement du fichier et écriture des données
190-260	Lecture des données
270-310	Edition de contrôle
320-360	Ajoût de données
370-420	Lecture de la disquette
430	Impression de contrôle après ajoût.
440	Fin du programme

Nous avons un peu raccourci la description de ce programme, car les détails sur la manipulation d'un fichier ont été expliqués dans les exemples précédents. Ici l'instruction correspondante est APPEND nom de fichier, qui ouvre non seulement le fichier (donc remplace l'instruction OPEN) mais également s'occupe d'ajouter les informations supplémentaires.

Si l'on veut maintenant s'occuper de données isolées (pour les détruire ou les modifier par exemple) c'est possible grâce à l'accès direct aux fichiers. Comme dans l'exemple suivant :

```
30  PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40  PRINT:PRINT:PRINT TAB(9) "R PRUST / W.VOSS 1984"
50  PRINT:PRINT:PRINT TAB(8)"2.5-MODIFICATION DU FICHIER"
```

```
60  PRINT:PRINT:PRINT:PRINT
70  DIM N$(4),B$(4)
80  PRINT "CE PROGRAMME MONTRE COMMENT ON"
90  PRINT "MODIFIE UNE DONNEE"
105 PRINT:PRINT"ON DONNE LES NOMS:":PRINT:PRINT "LEON,LOUIS,REMI,EMILE"
107 PRINT: PRINT"ON VEUT CHANGER REMI EN EVA"
108 PRINT: PRINT TAB(14)"UN INSTANT SVP": FOR I=1 TO 8000: NEXT I
110 DATA LEON,LOUIS,REMI,EMILE
120 FOR I=1 TO 4:READ N$(I): NEXT I
130 D$=CHR$(4)
140 PRINT D$; "OPEN FICHIER4,L10"
150 FOR I=1 TO 4:PRINT D$;"WRITE FICHIER4,R";I
160 PRINT N$(I):NEXT I
180 PRINT D$ ;"CLOSE FICHIER4"
190 REM LECTURE DE LA DISQUETTE
210 PRINT D$;"OPEN FICHIER4,L10"
220 FOR I=1 TO 4:PRINT D$;"READ FICHIER4,R";I
240 INPUT A$: B$(I)=A$
250 NEXT I
260 PRINT D$;"CLOSE FICHIER4"
270 REM IMPRESSION DE CONTROLE
280 HOME:PRINT "CONTROLE :":PRINT:PRINT
290 FOR I=1 TO 4
300 PRINT B$(I)
310 NEXT I
320 REM MODIFICATION
330 PRINT D$;"OPEN FICHIER4,L10"
340 PRINT D$; "WRITE FICHIER4,R";3
350 PRINT "EVA"
360 PRINT D$;"CLOSE FICHIER4"
370 REM CONTROLE
```

```
380 PRINT D$;"OPEN FICHIER4,L10"
390 FOR I=1 TO 4
400 PRINT D$;"READ FICHIER4,R";I
410 INPUT A$: B$(I)=A$: NEXT I
420 PRINT D$;"CLOSE FICHIER4"
430 PRINT:PRINT:PRINT"CONTROLE APRES MODIFICATION":PRINT
440 FOR I=1 TO 4: PRINT B$(I) : NEXT I
450 PRINT : PRINT"FIN": END
```

Description du programme:

phrases	contenu
30-107	Titre,dimensionnement et commentaires
108	Boucle d'attente
110-120	Entrée des données
130-180	Mémorisation
190-310	Lecture et impression de contrôle
320-360	Modification du contenu du fichier
370-440	Lecture et impression de contrôle
450	Fin

On imagine facilement comment on peut détruire des données isolées.

2.3.4 Exemple: Evaluations statistiques

Comme on l'a déjà dit un micro-ordinateur présente des avantages en particulier pour le traitement et l'analyse de la plupart des données existantes. Les travaux que les statisticiens effectuaient à la main ou en utilisant de gros ordinateurs chers et souvent surchargés sont maintenant maîtrisés rapidement et sans erreurs avec de petits ordinateurs.

Pour une telle évaluation de données, il faut installer l'ordinateur de manière à analyser les données existantes de la façon la plus commode possible et avoir la possibilité d'écarter les erreurs qui pourraient s'y glisser.

Afin d'illustrer ceci, nous allons utiliser les résultats d'un sondage. Un tel exemple s'impose parce que l'analyse de données plus importantes n'est possible en fait qu'avec la mise en oeuvre d'une mémoire périphérique. Il faudra donc nécessairement implémenter un lecteur parce que les données à exploiter doivent être disponibles en permanence ou du moins pour un bon moment.

Voyons les étapes nécessaires à ce travail:

Définition des problèmes:

On relève une série de données numériques par un sondage, par exemple:

1.Age (en années)
2.Taille en centimètres
3.Poids en kilos
4.Tension la plus élevée

5.Consommation de cigarettes (par jour)

Il faut étudier les points suivants :

1. Entrée des données sur disquette
2. Correction éventuelle des données
3. Edition de contrôle de tous les jeux de données
4. Edition alternative des lignes ou colonnes de la matrice de données selon les voeux de l'utilisateur
5. Définition de la valeur moyenne et des écarts types pour les variables particulières selon les voeux de l'utilisateur
6. Calcul du coefficient de corrélation

Méthodes statistiques

Pour les méthodes d'exploitation statistique on notera les points suivants:

La moyenne (moyenne arithmétique) d'une variable, se calcule de la manière suivante :

$$\bar{X}=1/n \sum_{i} Xi$$

avec $\bar{X}$ =moyenne arithmétique

Xi =valeur remarquable no i de la variable X

i =index

n =nombre d'individus

Un écart-type comme taux de dispersion de la valeur remarquable se calcule de la manière suivante :

$$s=\sqrt{1/n \quad (Xi-\bar{X})^2}$$

où s=écart type

Le coefficient de corrélation mesure l'intensité de la relation entre deux variables x et y :

$$r=\frac{n\sum_i XiYi-\sum_i Xi\sum_i Yi}{\sqrt{(n\sum_i Xi-(\sum_i Xi))(n\sum_i Yi-(\sum_i Yi))}}$$

où r=coefficient de corrélation
d'après BRAVAIS/PEARSON
(-1 =r =+1)

Les données

Pour pouvoir continuer notre illustration nous avons besoin de données:

Individus	Age	Taille	Poids	Tension	Cigarettes
1	20	182	80	142	30
2	30	174	72	134	0
3	25	178	80	140	20
4	45	168	68	132	10
5	42	177	75	130	0
6	58	169	70	145	20
7	37	176	82	150	20

Naturellement dans la pratique on a affaire en général à un ensemble de données plus important mais cela ne change rien à la problématique de base. Dans les pages suivantes nous établissons un programme qui viendra à bout des problèmes posés.

Dans ce programme il saute aux yeux que chaque jeu de données particulier est aussitôt mémorisé de manière à ce que l'utilisateur du programme ait la possibilité de corriger d'éventuelles erreurs lors de l'entrée.
Cette mémorisation directe a un sens profond : si l'on voulait entrer tous les jeux de données pour pouvoir les mémoriser,il y aurait un risque de perte des données (par exemple une panne de courant) incomparablement élevé.

On remarque que le premier jeu de données correspondant à l'instruction OPEN, doit être transmis plus tard à la disquette par l'instruction APPEND.

```
30 PRINT TAB(9)"APPPLE TRUCS ET ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R. PRUST / W. VOSS 1984"
50 PRINT : PRINT TAB(5)"2.7 - EVALUATION STATISTIQUE"
60 PRINT : PRINT
70 PRINT : PRINT : PRINT"CE PROGRAMME CONTIENT L'ETABLISSEMENT"
80 PRINT"DE DONNEES STATISTIQUES ET LEUR"
90 PRINT TAB(7)"EVALUATION STATISTIQUE"
95 PRINT : PRINT : PRINT"DANS CE PROGRAMME SEULES LES VARIABLES  NUMERIQUES
SONT": PRINT"PRISES EN CONSIDERATION"
97 PRINT : PRINT"LES MISSING VALUES NE SONT PAS PRISES EN CONSIDERATION"
100 GOSUB 1000 : REM ATTENDRE
110 INPUT "NOMBRE DE PERSONNES      :";N
120 PRINT : INPUT"NOMBRE DE VARIABLES     :";V
130 DIM W(N,V),X(N),Y(N)
140 FOR I=1 TO N
150 PRINT : PRINT : PRINT
160 PRINT"PERSONNE NR. ";I: PRINT :PRINT
170 FOR J=1 TO V
180 PRINT TAB(5)"VAR.NR. ";J: INPUT " : "W(I,J)
190 NEXT J
200 PRINT : PRINT : PRINT"EST-CE CORRECT?";: INPUT"(O/N)";A$
210 IF A$="O" THEN 230
220 GOSUB 2000: REM CORRECTION
230 IF I=1 THEN GOSUB 3000: REM MEMOIRE 1
240 GOSUB 4000: REM MEMOIRE 2
250 NEXT I
260 HOME : PRINT "EDITION DE CONTROLE": PRINT :PRINT
270 PRINT"INDIVIDU";  TAB(8)"AG";  TAB(13)"TA";  TAB(18)"PO";  TAB(23)"TE";
TAB(28)"CI"
300 IF I/10= INT(I/10) THEN GOSUB 1000: REM ATTENDRE
310 NEXT I
```

```
320 GOSUB 1000: REM ATTENDRE
330 PRINT"CONTROLE D'UN INDIVIDU ?": INPUT " (O/N)";A$
340 IF A$="N" THEN 400
350 PRINT : PRINT"QUEL INDIVIDU ?" (SVP NUMERO)";: INPUT I
360 GOSUB 5000 : REM INDIVIDU
370 GOTO 330
400 HOME : PRINT"CONTROLE COLONNE ?"INPUT" (O/N)";A$
410 IF A$="N" THEN 500
420 PRINT : PRINT"QUELLE COLONNE ?" (SVP NUMERO)";: INPUT J
430 GOSUB 6000: REM TABLEAU
440 GOTO 400
500 REM OPERATIONS STATISTIQUES
505 PRINT : PRINT : PRINT : PRINT
510 PRINT"OPERATIONS STATISTIQUES ? (O/N)";: INPUT A$
520 IF A$="N" THEN 900
530 PRINT : PRINT : PRINT"VALEUR MOYENNE ? (O/N)";: INPUT A$
540 IF A$="N" THEN 600
550 PRINT : PRINT : INPUT"QUELLE VARIABLE ? (SVP NUMERO)";J
560 FOR I=1 TO N:X(I)=W(I,J): NEXT I: GOSUB 7000: REM MOYENNE
570 GOSUB 1000: REM ATTENDRE
580 GOTO 530
600 HOME : PRINT"ECART-TYPE ?" (O/N);: INPUT A$
610 IF A$="N" THEN 700
620 PRINT : PRINT : PRINT"QUELLE VARIABLE ? (SVP NUMERO)";: INPUT J
630 FOR I=1 TO N:X(I)=W(I,J): NEXT I
640 GOSUB 7000: REM MOYENNE
650 GOSUB 8000: REM SIGMA
660 GOTO 600
700 HOME : PRINT"CORRELATIONS ? (O,N)"; INPUT A4
710 IF A$="N" THEN 900
720 PRINT : PRINT : PRINT"DONNEZ UNE VARIABLE SVP: ": PRINT : PRINT
```

```
730 INPUT"PREMIERE VARIABLE    :";J1
740 PRINT : INPUT"DEUXIEME VARIABLE    :";J2
750 FOR I=1 TO N:X(I)=W(I,J1)=W(I,J2):NEXT I
760 GOSUB 9000: REM CORRELATION
770 GOTO 700
900 PRINT : PRINT : PRINT"FIN DU PROGRAMME":END
1000 REM SP ATTENDRE
1010 VTAB 23: HTAB 6: INVERSE: PRINT"SVP APPUYER SUR UNE TOUCHE!":NORMAL
1020 GET A$: IF A$=" " THEN 1020
1030 HOME : RETURN
2000 REM SP CORRECTION
2010 PRINT: PRINT "QUEL ENONCE EST FAUX?"
2020 INPUT"DONNEZ UN NR SVP";J
2030 PRINT : PRINT TAB(5)"VALEUR ERRONEE:";W(I,J)
2040 PRINT : PRINT TAB(5)"VALEUR CORRECTE:";: INPUTW(I,J)
2050 HOME : PRINT"CONTROLE": PRINT: PRINT :PRINT"NR D'INDIVIDU";I :  PRINT:
PRINT2060 FOR J=1 TO V: PRINT"VAR. NR";J;TAB(15)W(I,J): NEXT J
2070 PRINT: PRINT"TOUT EST-IL CORRECT ?(O/N)";: INPUT A$
2080 IF A$="N" THEN PRINT: GOTO 2010
2090 RETURN
3000 REM SP MEMOIRE 1
3010 D$=CHR$(4)
3020 PRINT D$;"OPEN DATE15"
3030 PRINT D$;"WRITE DATE15"
3040 FOR J=1 TO V: PRINT W(I,J): NEXT J
3050 PRINT D$;"CLOSE DATE15"
3060 RETURN
4000 REM SP MEMOIRE 2
4010 D$=CHR$(4)
4020 PRINT D$;"APPEND DATE15"
4030 PRINT D$;"WRITE DATE15"
```

```
4040 FOR J=1 TO V: PRINT W(I,J): NEXT J
4050 PRINT D$;"CLOSE DATE15"
4060 RETURN
5000 REM SP LIGNE
5010 HOME : PRINT"INDIVIDU NR";I: PRINT : PRINT
5020 FOR J=1 TO V: PRINT"VAR. NR";J;TAB(15)W(I,J): NEXT J
5030 GOSUB 1000: REM ATTENDRE
5040 RETURN
6000 REM SP COLONNE
6010 HOME : PRINT"COLONNE NR";J: PRINT :PRINT
6020 FOR I=1 TO N: PRINT"IDIVIDU NR";I;TAB(15)W(I,J) : NEXT I
6030 GOSUB 1000 : REM ATTENDRE
6040 RETURN
7000 REM SP MOYENNE
7010 S=0
7020 FOR I=1 TO N: S=S+X(I): NEXT I
7030 AM=S/N
7040 PRINT: PRINT: PRINT"VALEUR MOYENNE DE LA VARIABLE NR ";J;"
:";INT(AM*100+.5)/100
7060 RETURN
8000 REM SP SIGMA
8010 S=0
8020 FOR I=1 TO N: D=X(I)-AM: D=D*D : S=S+D: NEXT I
8030 SA=SQR(S/N) : S=INT(S*100+.5)/100
8040 PRINT : PRINT :PRINT"ECART-TYPE :";SA
8050 GOSUB 1000: REM ATTENDRE
8060 RETURN
9000 REM SP CORRELATION
9010 S1=0 : S2=0 : S3=0: S4=0 : S5=0
9020 FOR I=1 TO N : S1=S1+X(I)*Y(I) : S2=S2+X(I): S3=S3+Y(I) :
S4=S4+X(I)*X(I): S5=S5+Y(I)+Y(I) :NEXT I
```

```
9030 Z=N*S1-S2*S3
9040 NN=SQR((N*S4-S2*S2)*(N*S5-S3*S3))
9050 R=Z/NN: R=INT(R*1000+.5)/1000
9060 PRINT: PRINT :PRINT"CORRELATION ENTRE"
9070 PRINT : PRINT"VAR. NR.";J1;" ET "
9080 PRINT : PRINT"VAR. NR.";J2;" = ";R
9090 GOSUB 1000 : RETURN
```

Pour la description du programme, nous avons compartimenté les différentes parties du programme, cela permet au lecteur d'étudier ses particularités (c'est un exercice important, qui permet de s'entraîner et d'améliorer ses connaissances en BASIC)

Description du programme

Lignes	Contenu
30-97	Titre et commentaire
100-250	Entrée des données, correction des erreurs et mémorisation sur disquette
260-320	Edition de contrôle
330-370	Sortie d'un jeu de données
400-440	Sortie d'un tableau
500-580	Calcul de la valeur moyenne
600-660	Calcul de l'écart type
700-770	Calcul du coefficient de corrélation
900	Fin de programme
1000-1030	SP attendre
2000-2090	SP correction des données erronées
3000-3060	SP mémorisation par OPEN
4000-4060	SP mémorisation par APPEND
5000-5040	SP sortie d'un jeu de données
6000-6040	SP sortie d'un tableau
7000-7060	SP calcul d'une valeur moyenne
8000-8060	SP écart-type
9000-9090	SP coefficient de corrélation

Pour que le lecteur puisse bien évaluer ce que ce programme représente, des données quelconques ont été attribuées plus haut.

Résultats :

Après l'entrée des données, le programme effectue une édition de contrôle des données. Ensuite on peut contrôler toutes les valeurs moyennes, les écarts-types et les coefficients de corrélation. Ce qui donne:

	Valeur moyenne	Ecart-type
1.AGE	36.71	12.02
2.TAILLE	174.86	4.61
3.POIDS	75.29	5.09
4.TENSION	139	6.78
5.CIGARETTES	15	11.02

Coefficients de corrélation:

	1 Age	2 Taille	3 Poids	4 Tension	5 Cigarettes
1 Age	1	-0.825	-0.664	-0.005	-0237
2 Taille		1	0.848	0.16	0.337

3 Poids	1	-0.542	-0.573
4 Tension		1	0.831
5 Cigarettes			1

Il est facile de s'apercevoir qu'un programme aussi important peut être amélioré et aménagé de plusieurs points de vue. Quelques uns de ces aspects vont être décrits ici.

1. Lorsque les données sont mémorisées, comme nous l'avons vu plus haut, on peut naturellement prévoir que lors des évaluations statistiques toutes les informations ne pourront être stockées en mémoire mais les tableaux devront être lus sur disquette.

Il est facile de réaliser un programme de lecture correspondant.

2. Pour éviter d'éventuelles "MISSING VALUE" on questionnera le IF et on réduira le nombre de vraies réponses.

3. On n'a pas toujours affaire à des variables numériques: si des chaînes sont traitées elles ne pourront être dimensionnées avec W(N,V) mais ces variables devront être dimensionnées avec un tableau indépendant.

Les boucles de J=1 à V doivent être modifiées en conséquence.

4. Si on désire rendre le résultat visible et préciser le contenu des descriptions (Age,taille,...)

On devra placer ces chaînes dans un tableau B$(J) (J=1 à V) et elles devront être lues avec DATA et affichées au bon endroit.

Ces remarques doivent faire comprendre qu'un programme de traitement de données demande beaucoup d'attention si l'on désire obtenir des programmes confortables.

Chapitre 3: Accès mémoire et langage machine

3.1 Accès direct à la mémoire

Les éléments de BASIC dont nous avons parlé dans les deux premiers chapitres concernent, dans la plupart des cas, des problèmes inhérents à la programmation.

Pour un utilisateur qui est déjà avancé (ou en progrès) il ne s'agit plus de trouver n'importe quelle solution à son problème, mais d'autres aspects entrent en jeu: l'élégance de la programmation, la vitesse de traitement,....Dès qu'on est pris par la fièvre informatique, on cherche les meilleures solutions.

Il existe des styles de programmations différents comme il existe différents langages. Ce chapitre ne constitue qu'une base où les performances du programme ne seront pas mises en avant.
Si différentes soient les conceptions d'un bon programme, on s'efforcera ,pour élaborer le meilleur programme de suivre sa propre conception.

Jusque là nous avons laissé à l'ordinateur le soin de gérer sa mémoire.

Le contenu de la mémoire n'est questionné que par des commandes comme LIST et PRINT.
L'APPLE possède néanmoins la possibilité de questionner directement le contenu d'un emplacement mémoire et même de le modifier.

Pour que cela soit possible on décidera de la valeur précise à mettre dans

l'emplacement désiré.
Quelques possibilités vont être exposées par la suite. Nous ne dresserons pas la liste complète des possibilités.

C'est la commande

POKE, Adresse, Valeur

qui permet d'envoyer une valeur à l'adresse donnée.

La fonction

PEEK (Adresse)

permet de lire le contenu de l'adresse indiquée.

Dans les deux cas la valeur doit être donnée en décimal.

Exemples:

```
POKE 33,30
POKE 32,5
PRINT PEEK(222)
PRINT PEEK(103)+PEEK(104)*256
```

les deux premiers POKE permettent de réduire la largeur de l'écran à 30, puis la colonne de début est fixée à 5.

Les parties gauche et droite de l'écran sont donc fermées et ne sont plus accessibles.
La commande HOME n'est valable qu'à l'intérieur de la fenêtre.

PEEK doit être combinée comme toute fonction avec PRINT lorsque le résultat doit être visible.

Comme on le voit elle peut être employée dans une expression arithmétique.

L'adresse 222 contient le code de la dernière erreur. Dans cet exemple un nombre en décimal est donné dans la fonction PEEK.

Le deuxième exemple convertit les deux parties d'un nombre hexadécimal. Les adresses 103 et 104 contiennent l'adresse à laquelle le programme actuel commence; c'est généralement 2049.

1. Applications
Lorsqu'on entre un nouveau programme, la mémorisation commence en général à l'adresse 2049 ($0801) en hexadécimal. La partie RAM (jusqu'à $0800) qui

est située en-dessous de cette adresse est réservé au système d'exploitation. En 103 on trouvera la partie basse de l'adresse de début de programme ($08=1 en décimal. En 104 on trouvera la partie haute ($08=8 en décimal). La formule déjà cité permetteant de déterminer le début de programme donne 2049.

Le prochain emplacement mémoire libre après la fin du programme est donné par les adresses 176 et 175 . L'ordre PRINT PEEK (175)+PEEK (17)*256 donne la première adresse libre après le programme .

On peut se servir de cette adresse pour protèger son programme.

Les opérations se déroulent de la façon suivante :

le lecteur s'assure avec LIST que le programme se trouve dans la mémoire (autrement il faut en mémoriser un ou le taper).

Ensuite on donne l'ordre suivant en mode direct :

```
POKE 103,PEEK(175):POKE 104, PEEK(176)
POKE PEEK(103)+256*PEEK(104)-1,0
```

Si maintenant on entre LIST, on constate que le programme a disparu. Le programme entré auparavant est visible à nouveau avec :

```
POKE 103,1 : POKE 104,8.
```

On trouvera des exemples dans le chapitre 7

2. La fenêtre de l'écran a déjà été évoquée lors de la présentation des instructions. Il est possible de réduire l'écran pour améliorer une présentation. Ceci est intéressant surtout lorsque des parties d'écran doivent être protégées contre des impressions indésirables. Les emplacements de mémoire qui sont utilisables pour cela vont de 32 à 35.

32 donne la rangée gauche de l'écran. Les valeurs de l'adresse 32 doivent être comprises entre 0 et 39.

33 donne la largeur de l'écran. Les valeurs permises sont situées entre 1 et 40, il faut faire attention à ce que PEEK (33) + PEEK (32) ne soient pas plus grand que 40. Il faut modifier 33 puis 32.

34 donne la valeur de la ligne du haut de l'écran (comprise entre 0 et 23). Indique la ligne à supprimer.

35 correspond à la dernière ligne de l'écran (valeur comprise entre 0 et 23).

Les possibilités d'utilisation sont nombreuses.
Nous aurions pu programmer l'en-tête de tous nos programmes de la manière suivante:

```
3 HOME : HTAB(9):PRINT "APPLE-TRUCS ET ASTUCES"
4 PRINT:PRINT:HTAB (9):PRINT"R.PRUST/ W.VOSS 1984":PRINT
5 FOR I=1 TO 40 : PRINT"-": NEXT:POKE 34,5:HOME
```

Il reste encore 18 lignes pour une éventuelle impression. Avec POKE 34,6 les lignes 0 jusqu'à 5 sont fermées sur l'écran . La commande HOME n'efface que l'intérieur de la fenêtre d'écran.

On peut faire quelque chose de semblable avec les lignes du bas lorsque dans les dernières lignes le nom du programme sur lequel on vient de travailler doit rester visible pendant son déroulement. On peut développer un programme de la manière suivante:

```
3 HOME: HTAB (9) :PRINT "APPLE-TRUCS ET ASTUCES"
4 PRINT :PRINT : HTAB (9):PRINT "R.PRUST/W.VOSS 1984" PRINT
```

```
5 FOR I= 1 TO 40: PRINT "-";:NEXT: POKE 34,6: HOME
10 VTAB 22: FOR I=1 TO 40:PRINT "-";:NEXT:PRINT:HTAB 9:PRINT "PROGRAMME
DEMO TEXTE DE FENETRE";:POKE 35,21:HOME.
```

La dernière ligne disponible est maintenant la ligne 20, c'est à dire qu'il nous reste encore 15 lignes.
Les limites de l'écran sont fixées après le premier ordre qui s'adresse à un nouvel emplacement de mémoire, c'est pourquoi l'instruction HOME suit les lignes 5 et 10. On s'assure par ce moyen que les instrutions PRINT suivantes ne seront positionnés qu'à l'intérieur de la fenêtre.

Maintenant le programme va être développé pour quelques commandes d'impression et des INPUT. On demande cinq entrées pour les variables A1 à A5 dont la moyenne sera ensuite calculée. L'impression sera tabulée à la colonne 6. On y parvient avec TAB(6) dans le programme suivant :

```
3 HOME : HTAB (9) : PRINT"APPLE-TRUCS ET ASTUCES"
4 PRINT:PRINT:HTAB(9):PRINT "R.PRUST/W.VOSS 1984":PRINT
5 FOR I=1 TO 40 : PRINT "-";:NEXT:POKE 34,6:HOME
10 VTAB 22:FOR I=1 TO 40:PRINT"-";:NEXT:PRINT:HTAB 9:PRINT "PROGRAMME:DEMO
TEXTE DE FENETRE";:POKE 35,21:HOME
20 PRINT TAB (6)" 1.ENTREE:";:INPUT" " ";A1
30 PRINT TAB (6) "2 ENTREE:";:INPUT " ";A2
40 PRINT TAB (6) "3 ENTREE:";:INPUT " ";A3
50 PRINT TAB (6) "4 ENTREE:";:INPUT " ";A4
60 PRINT TAB (6) "5 ENTREE:";:INPUT " ";A5
70 X=(A1+A2+A3+A4+A5)/5
80 PRINT :PRINT:PRINT TAB(6) " MOYENNE :";X
90 END
```

Lorsque de très nombreux déplacements doivent se suivre il est plus

pratique de changer la place de la colonne de gauche plutôt que d'utiliser la fonction TAB.

```
3 HOME : HTAB(9:PRINT"APPLE-TRUCS ET ASTUCES"
4 PRINT:PRINT:HTAB(9):PRINT"R.PRUST/W VOSS 1984":PRINT
5 FOR I=1 TO 40:PRINT "-";:NEXT:POKE 346:HOME
10 VTAB 22:FORI=1 TO 40:PRINT "-";:NEXT:PRINT:HTAB 9:
PRINT"PROGRAMME:DEMO TEXTE DE FENETRE";:POKE 35,21:
HOME
20 POKE 33,35:POKE 32,5 : HOME
30 INPUT "1.ENTREE:";A1
40 INPUT "2.ENTREE:";A2
50 INPUT "3.ENTREE:";A3
60 INPUT "4.ENTREE:";A4
70 INPUT "5.ENTREE:";A5
80 X=(A1+A2+A3+A4+A5)/5
90 PRINT:PRINT:PRINT"MOYENNE:";X
100 END
```

Le résultat est identique à celui de la version précédente, mais le programmateur a économisé du temps de frappe, le programme est un peu plus court et lors de l'exécution on supprime six ordres TAB c'est à dire que l'on économise du temps.
Ceci se fait naturellement surtout pour des programmes plus importants.

Voici une autre utilisation de POKE 32,X ,la partie gauche de l'écran doit être décrite avant la partie droite.
Ainsi on peut placer six variables les unes à côté des autres:trois à gauche et ensuite trois à droite. Voici ce qui arrive :

```
3 HOME:HTAB(9):PRINT"APPLE-TRUCS ET ASTUCES"
```

```
4 PRINT:PRINT:HTAB(9):PRINT"R.PRUST/W.VOSS 1984":PRINT
5 FOR I=1 TO 40 :PRINT"-";:NEXT:POKE 34,6:HOME
10 VTAB22:FOR I = 1 TO 40 : PRINT "-";:NEXT: PRINT: HTAB9: PRINT"PROGRAMME:
DEMO, TEXTE DE FENETRE";: POKE 35,21 : HOME
20 INPUT "1 ENTREE :";A1
30 INPUT "2 ENTREE :";A2
40 INPUT "3 ENTREE :";A3
50 POKE 33,20:POKE 32,20 :HOME
60 INPUT "4 ENTREE :";A4
70 INPUT "5 ENTREE :";A5
80 INPUT "6 ENTREE :";A6
90 PRINT "SOMME:";A1+A2+A3+A4+A5+A6
100 END
```

Les lignes 20 à 40 travaillent comme d'habitude, le texte commence sur la colonne gauche de l'écran. La largeur de l'écran est réduite de 20 avec la ligne 50 et la colonne gauche est placée en position 20.

Tout d'abord la largeur est réduite, ensuite la rangée placée. La commande HOME a de nouveau la fonction déjà décrite, les instructions précédentes POKE sont de nouveau valables.

Les messages des lignes de 60 à 80 sont placés à côté des lignes précédentes (des lignes 20 à 40). L'impression du résultat de la ligne 90 est également située dans la partie droite de l'écran à partir de la colonne No 20 (c'est à dire la colonne 21).

On peut restituer l'écran original en utilisant la commande POKE avec les valeurs d'origine. Un dernier exemple va montrer ceci; il va de soi qu'il

faut élargir de nouveau l'écran avant l'impression du résultat pour que la somme soit calculée avec les 6 entrées.

```
3 HOME : HTAB (9):PRINT "APPLE TRUCS ET ASTUCES"
4 PRINT:PRINT : HTAB (9) :PRINT "R.PRUST/W.VOSS 1984":PRINT
5 FOR I =1 TO 40 :PRINT "-";: NEXT :POKE 34,6 : HOME
6 10 VTAB 22 : FOR I=1 TO 40 : PRINT "-";:NEXT:PRINT:HTAB (9):PRINT
"PROGRAMME:DEMO TEXTE DE FENETRE ";:POKE 35,21:HOME
20 INPUT "ENTREE 1 :" ; A1
30 INPUT "ENTREE 2 :" ; A2
40 INPUT "ENTREE 3 :" ; A3
50 POKE 33,20 : POKE 32,20 : HOME
60 INPUT "ENTREE 4 :" ; A4
70 INPUT "ENTREE 5 :" ; A5
80 INPUT "ENTREE 6 :" ; A6
90 PRINT:PRINT
100 POKE 32,0 :POKE 33,40
110 PRINT : PRINT "LA SOMME S'ELEVE A :" ;A1+A2+A3+A4+A5+A6
120 END
```

Après les fonctions déjà connues des lignes 3 à 80 , deux lignes vides sont intercalées (seulement sur la partie droite de l'écran).
Ensuite les lignes de l'écran et leur largeur sont rétablies à leurs valeurs normales 0 et 40 .
La première commande PRINT de la ligne 110 ne s'applique encore qu'à la partie droite de l'écran; c'est pourquoi il faudrait établir l'impression du résultat avant , pour qu'ensuite elle commence de nouveau sur le côté gauche de l'écran.

Une troisième possibilité d'application concerne l'écran mais pas la fenêtre de l'écran:

```
3 HOME : HTAB (9):PRINT "APPLE TRUCS ET ASTUCES"
4 PRINT:PRINT : HTAB (9) :PRINT "R.PRUST/W.VOSS 1984":PRINT
10 HOME : FOR J = 1 TO 24 : FOR I = 1 TO 40 : PRINT "A";:NEXT I,J
20 FOR Q = 1 TO 2000 : NEXT
30 X = 160
40 FOR Q = 1024 TO 2039 : POKE Q,X :NEXT
50 END
```

Le programme remplit en premier lieu l'écran entier avec la lettre "A" en ligne 10. La ligne 20 contient une boucle d'attente qui compte jusqu'à 2000 et qui sert à ce que l'on puisse regarder plus longtemps l'écran . Avec la boucle de la ligne 40, l'écran est de nouveau effacé (d'abord la ligne 0, ensuite la ligne 8, puis la ligne 16, suivies des lignes 1,9,17, puis des lignes 2,10,18, etc...). Avec POKE Q,X des espaces sont écrits dans tous les emplacements de mémoire de 1024 à 2039 (il s'agit de la mémoire-écran). Dans les emplacements de mémoire de 1024 jusqu'à 1063 il y a les codes de tous les caractères de la première ligne, de 1064 à 1103 ceux de la neuvième ligne, de 1104 jusqu'à 1143 ceux de la 17ème ligne.
Ensuite on "saute" 8 emplacements et les emplacements de 1152 à 1191 contiennent tous les codes des caractères de la deuxième ligne etc...
La partie de l'écran est ainsi divisée en trois groupes : ligne 0 à 7, ligne 8 à 15 , et ligne 16 à 23. Le stockage du contenu se fait alors de la manière suivante : une ligne du premier groupe, une du second, une du troisième , ensuite 8 emplacements pour d'autres informations, ensuite de nouveau une ligne du premier groupe , une du second, une du troisième , 8 emplacements "libres" etc...

Dans le premier exemple on utilise l'accès mémoire direct pour effacer

l'écran entier.On peut néanmoins naturellement mettre un caractère avec la commande POKE à un endroit bien précis, et faire commencer une suite de caractères à un emplacement précis. Ceci sera montré dans le programme suivant, qui présentele même écran que le programme précédent des lignes 20 à 110.

```
3 HOME : HTAB (9):PRINT "APPLE TRUCS ET ASTUCES"
4 PRINT:PRINT : HTAB (9) :PRINT "R.PRUST/W.VOSS 1984":PRINT
5 FOR I=1 TO 2500: NEXT
10 HOME:PRINT:A$="ENTREE:":L=LEN(A$)
20 A=1664:E=3:I=1
30 FOR QQ=0 TO 1
40 Q=A:F$=STR$(I)
50 POKE Q,ASC (F$)+128
60 FOR Z=1 TO L: Q=Q+1: F$=MID$ (A$,Z,1):POKEQ,ASC (F$)+128 : NEXT
70 POKE Q+1,96
80 POKE 37,0: PRINT
90 INPUT "";X:S=S+X
100 B$=STR$ (X):LX=LEN (B$)
110 FOR Z=1 TO LX:Q=Q+1: F$=MID$ (B$,Z,1): POKE Q,ASC(F$+128): NEXT
120 POKE 37,0
130 A=A+128: I=I+1: IF I =E THEN 40
140 A=1684: E=6
150 NEXT QQ
160 A$="LA SOMME COMPREND: ": L=LEN(A$)
170 Q=1447
180 FOR Z=1 TO L: Q=Q+1: F$=MID$(A$,Z,1): POKE Q,ASC(F$)+128: NEXT
190 A$=STR$(S): L=LEN(A$)
200 FOR Z=1 TO L: Q=Q+1: F$=MID$(A$,Z,1): POKE Q,ASC(F$)+128: NEXT
210 FOR I=1024 TO 1036: POKE I,160: NEXT
220 END
```

Description du programme

Phrases	Contenu
10 à 20	On attribue les valeurs de début des variables A$ contient la partie permanente de l'édition, la longueur est mémorisée en L La variable A tient lieu d'adresse, sa valeur de début s'élève à 1664; ceci est l'emplacement de mémoire pour le premier caractère de la sixième lignes de l'écran (ligne numéro 5) . L'impression doit commencer ici. E correspond au dernier numéro d'une entrée à gauche ou à droite de l'écran, au départ E est égal à 3 I est le compteur des entrées et commence donc à la valeur 1.
30	C'est le début d'une boucle qui permet de distinguer les 2 parties de l'écran.
40	Q est l'emplacement actuel de mémoire , on lui attribue au début l'adresse A. F$ est la chaîne qui contient les caractères pour le nombre d'entrée I
50	Le code écran des caractères de la chaîne F$

(ici numéro d'entrée) est écrit à l'emplacement Q (ici adresse de début A)

60 Avec la suite d'instructions les codes de tous les caractères de la chaîne A$ (ici A$ = "ENTREE") seront écrits dans les emplacements de mémoire successifs (c'est pourquoi Q est incrémenté) S'il doit s'agir des caractères sur une ligne les adresses correspondantes sont situées directement les unes derrière les autres

70 suite au texte, comme dans un IMPUT, un carré lumineux en inverse doit apparaître. Il a le code 96 et sera donc écrit dans le prochain emplacement de mémoire

80 pour que le curseur ne s'en aille pas vers le bas de l'écran avec la commande INPUT de la ligne 90 il est positionné à la première place en haut à gauche avec POKE 37,0, le PRINT suivant ne concerne que l'initialisation de la commande POKE.

90 L'entrée d'une valeur suit maintenant dans le coin supérieur gauche de l'écran .

100-110 A la place du carré lumineux, sont écrits maintenant tous les caractère du nombre attribué à X (éventuellement le point décimal également) dans la mémoire de l'écran. La suite des commandes à déjà été vue en ligne

60. Au lieu de A$ il faut employer la variable B$ qui est la chaîne engendrée par X, la valeur de A$ sera encore utilisée dans la suite du programme.

120 Le curseur est de nouveau placé vers le haut

130 A est modifié pour pouvoir changer de ligne
Pour que les prochaines lignes appartiennent au même groupe de lignes de l'écran, on ajoute 128 comme nous l'avons expliqué précédemment.
Pour cela le compteur passe à 1 et on teste si la dernière entrée dans cette partie de l'écran a eu lieu. Sinon on recommence le déroulement des lignes 40 à 130, on incrémente I, les valeurs de début sont modifiées ainsi que Q; cette dernière modification a lieu dans la ligne 40

140 Dans le cas ou I a atteint une valeur supérieure à E, les entrées sont faites sur l'autre partie de l'écran A prend la valeur 1684, qui est l'adresse pour le caractère 21 de la ligne 6. E prend pour valeur limite 6.
I prend la nouvelle valeur 4.

150 Toutes les instructions sont de nouveau répétées et cela pour l'inscription de la partie droite de l'écran donc de la quatrième à la

sixième entrée.

---

160 La variable A$ a pour valeur le nouveau texte et la variable L correspondant à sa longueur est de nouveau calculer

170 Q a pour valeur 1447 soit le dernier emplacement pour les premiers caractères de la ligne 12 (cela correspond aux instructions PRINT de l'avant dernier programme).

180 Le texte de A$ est écrit caractère par caractères dans l'emplacement de mémoire situé à partir de 1448 (car Q est passé à 1) l'indicateur d'écran ainsi établi commence aux premiers caractères ligne 12.
Avec la modification à la ligne 170 insérer l'impression en établissant Q= 1453.

190-200 La somme S calculée est écrite caractère par caractère dans les emplacements voulus

210 Si la première ligne de l'écran contient encore les chiffres de la dernière entrée (ligne 90 pour I =6), les 13 premiers emplacements de l'écran sont remplis par des espaces. Il faudrait aussi penser pour plus de sûreté à remplir toute la ligne avec des espaces.

Avec cela la valeur de la fin de la boucle

passe de 1036 à 1063.

220 Fin du programme.

Nous allons encore préciser quelques unes des instructions du programme précédent. Les lignes 80 et 120 contiennent une commande POKE pour l'emplacement de mémoire 37. La valeur de cet emplacement donne la position verticale et actuelle du curseur, donc de la ligne de l'écran dans laquelle le curseur se trouve.
Les valeurs permises se situent entre 0 et 23.
Encore une courte remarque pour les fonctions STR$ et ASC.
STR$ s'utilise comme opérande d'une expression numérique, c'est à dire un nombre, une variable numérique ou une formule arithmétique.
Avec cette fonction le nombre devient une chaîne c'est à dire qu'il est considéré par l'ordinateur comme une suite de caractères composant un nombre.
Grâce à cela on peut mettre en place les fonctions de traitement de chaînes LEFT$, RIGHT$ et MID$.

Ces fonctions sont utilisées dans le programme. ASC ne peut être employé qu'avec une chaîne.
Cette fonction transforme le premier caractère de la chaîne en code ASCII correspondant.
Comme elle ne peut "traduire" qu'un signe à la fois , il faut revenir dans notre programme plusieurs fois grâce à des boucles et des fonctions MID$ (A$,Z,1).
Il va de soi que la fonction ASC n'a pas qu'une signification. Elle permet de POKER le code écran en ajoutant 128 à la valeur donnée par ASC. En effet le code ASCII correspond au code clavier

Un exemple : la commande PRINT ASC (" ") donne le code ASCII de l'espace 32

POKE 1192,32 permet d'écrire au premier emplacement de la ligne 10 de l'écran. Ce n'est pas un espace qui sera visible, mais un espace inverse. L'écran peut contenir trois types d'espaces: un normal, un en inversé et un clignotant.
Ceci est valable pour tous les caractères.

D'ordinaire l'ordinateur reçoit une affectation, qu'elle soit donnée avec la commande INVERSE ou FLASH, il faut additionner 128 à tous les codes qui viennent du clavier. Si par contre c'est INVERSE qu'on utilise, le code peut être accepté directement. Si c'est FLASH qui est en vigueur l'ordinateur ajoute 64. Le mode de sortie actuel qui est indiqué dans l'emplacement de mémoire 50 est modifié.

60 correspond à INVERSE
127 correspond à FLASH
255 correspond à NORMAL

Lors de l'accès mémoire direct la gestion ne sera pas faite par l'ordinateur c'est pourquoi l'utilisateur doit prendre en compte ce problème. Vous trouverez dans les manuels APPLE les tableaux de codes correspondants.

Le dernier programme a rendu compréhensible l'utilisation de l'accès mémoire direct pour la disposition de l'écran, mais a également montré que dans quelques cas la programmation peut être réellement difficile . Si les lignes de l'écran décrites n'avaient pas appartenu au même groupe la difficulté aurait été plus grande. Néanmoins un tel programme donne un

aperçu de la façon de travailler de l''ordinateur et enfin de telles connaissances ouvrent la possibilité de maitriser des problèmes précis de programmation.

Le programme MASKEN GENERATOR du chapitre 7.4 est un très bon exemple d'application pour cela. Sans ces connaissances, l'utilisation de la mémoire écran ne serait pas pensable.

4. La quatrième possibilité d'application de l'accès direct mémoire est le clavier.
La connaissance de la mémoire clavier est utilisée relativement souvent pour l'établissement du type de problème suivant :

Le programme doit être interrompu pour donner à l'utilisateur l'occasion de regarder tranquillement l'écran. Ainsi les informations livrées par le programme peuvent être lisible le temps que l''utilisateur les lise avant de continuer son travail . Plusieurs tableaux de résultats peuvent se suivre les uns derrière les autres ,et là aussi une interruption est nécessaire , pour que les résultats puissent être lus, en particulier lorsque l'imprimante n'est pas installée. Le programme peut encore afficher des erreurs que l'utilisateur doit connaître pour la suite du déroulement du programme. Le programme peut aussi être interrompu en appuyant sur une touche.
Pour ces deux cas nous allons donner de petits exemples de programmes.

Si on souhaite arrêter un programme, jusqu'à ce que :l'utilisateur désire la suite, on peut naturellement utiliser la commande STOP à l'endroit du programme désiré. Après avoir entré CONT au clavier, le programme continue à l'endroit exact de l'interruption :

```
3 HOME : HTAB (9) : PRINT "APPLE TRUCS ET ASTUCES"
```

```
4 PRINT:PRINT:HTAB (9):PRINT "R.PRUST/W.VOSS 1984":PRINT
55 FOR I = 1 TO 2000 : NEXT
10 HOME : PRINT "CE PROGRAMME SERT DE DEMONSTRATION":
PRINT:PRINT "IL MONTRE DE QUELLE MANIERE" :PRINT:PRINT
"UN PROGRAMME INTERROMPU PEUT :PRINT:PRINT:"ETRE REMIS EN SERVICE"
20 PRINT:PRINT:PRINT:PRINT:PRINT "VOUS POUVEZ CONTINUER LE
PROGRAMME" :PRINT:PRINT "AVEC L'ENTREE DE";: INVERSE :PRINT "CONT",:
NORMAL:PRINT"SUR LE " : PRINT:PRINT "CLAVIER"
30 STOP
40 HOME:PRINT "I";"I*I":PRINT
50 FOR I = 1 TO 50
60 PRINT I,I*I:IF I/10  50 THEN NEXT I
80 PRINT "LE PROGRAMME EST TERMINE"
90 END
```

Un message indique à l'utilisateur la commande à taper pour continuer le programme. Après CONT l'ordinateur continue le déroulement du programme. Maintenant nous allons voir un programme simple qui affiche les nombres de 1 jusqu'à 50 et leurs carrés. Toutes les 10 lignes le programme sera de nouveau interrompu avec le message correspondant et sera remis en marche avec CONT.
Ce programme sert de démonstration , il indique de quelle manière il faut interrompre puis remettre en marche un programmme.
Vous pouvez remettre le programme en marche en tapant CONT au clavier.

```
CE PROGRAMME SERT DE DEMONSTRATION
IL MONTRE DE QUELLE MANIERE
UN PROGRAMME INTERROMPU PEUT
ETRE REMIS EN SERVICE
```

```
VOUS POUVEZ CONTINUER LE PROGRAMME
AVEC L'ENTREE DE CONT
AU CLAVIER

BREAK IN 30
]CONT

  I             I*I
1               1
2               4
3               9
4               16
5               25
6               36
7               49
8               64
9               81
10              100

BREAK IN 70
]CONT
LE PROGRAMME EST TERMINE
```

Les messages :

BREAK IN 30 ou BREAK IN 70

pertubent la présentation de l'écran; l'utilisateur doit savoir qu'il s'agit d'un message d'erreur. Pour finir nous allons voir d'autres exemples qui n'utilisent pas CONT.

Avec l'instruction GET, il est possible d'améliorer le programme en procédant comme suit:

```
3 HOME:HTAB (9):PRINT "APPLE TRUCS & ASTUCES"
4 PRINT:PRINT:HTAB (9) :PRINT "P.PRUST/W.VOSS 1984"PRINT
5 FOR I = 1 TO 2000: NEXT
10 HOME:PRINT "CE PROGRAMME SERT DE DEMONSTRATION": PRINT:PRINT "IL MONTRE
DE QUELLE MANIERE" :PRINT:PRINT "ON PEUT INTERROMPRE ET REMETTRE"
:PRINT:PRINT" EN MARCHE UN PROGRAMME"
30 GOSUB 200
40 HOME:PRINT "I",I*I":PRINT
50 FOR I=1 TO 500
60 PRINT I,I*I :IF I/10    INT (I/10) THEN NEXT I
70 GOSUB 200 :PRINT:PRINT:IF I  50 THEN HOME :PRINT"I"
"I*I":PRINT:NEXT I
80 VTAB 22: PRINT "LE PROGRAMME EST TERMINE"
90 END
200 VTAB 22 : HTAB 6 : INVERSE:PRINT "APPUYEZ SUR UNE TOUCHE": NORMAL
210 GET A$ : IF A$ = " " THEN 210
220 VTAB 22:CALL -958 :RETURN
```

Description du programme

Phrases	Contenu
10	Le texte explique à quoi sert le programme la ligne 20 du programme précédent a disparu les explications pour la manipulation ne sont pas nécessaires
30	Branchement dans le sous-programme en ligne 200 qui conduit à une interruption
40	L'édition de l'écran se prépare : l'écran s'efface et la ligne de titre est affichée une ligne vide suit
50	Début de boucle
60	Après l'édition des résultats on vérifie si 10 nombres sont déjà apparus Sinon cette édition apparaîtra dans les prochaines lignes
70	Branchement dans le sous-programme 200 là après 10 nombres une interruption doit avoir lieu Ensuite on vérifie s'il faut sortir le bloc suivant si oui l'écran s'efface à nouveau le le titre apparaît avec un saut de ligne

La boucle se poursuit.

80 Si I dans la ligne 70 est égal à 50, un message signalant la fin du programme apparaît en ligne 21 de l'écran.

90 Fin du programme

200-220 Sous-programme d'interruption

200 Dans la ligne de l'écran 22 le message apparaît , précisant comment continuer le programme en appuyant sur une touche
Le texte est en inverse et tabulé en colonne 5.
Pour finir on retourne à un mode d'édition NORMAL

210 Avec la commande GET le programme est interrompu aussi longtemps qu'une touche n'est pas enclenchée sur le clavier
Si la chaîne A$ est vide, on reboucle sur 210

220 Après avoir effacé la ligne 22 avec CALL -958 l'ordinateur retourne au programme principal.

Avec cette transformation du programme on a atteint deux buts : l'appui

d'une touche pour pouvoir continuer le déroulement du programme , et un écran non perturbé par les messages du système d'exploitation de l'ordinateur. Un seul message du programme suffit pour informer l'utilisateur de la manipulation. Néanmoins cette façon de programmer a un désavantage : si l'on appuie sur deux touches pour s'exercer, ou par ignorance, le deuxième caractère reste dans la mémoire du clavier et lors du GET suivant l'ordinateur n'attend plus de nouvelle entrée.

Le deuxième désavantage réside dans l'inexplicable interaction établie entre la commande GET et quelques commandes DOS.
On peut éviter ces deux problèmes en suivant les versions de programmes suivantes :

```
3 HOME:HTAB (9):PRINT "APPLE TRUCS & ASTUCES"
4 PRINT:PRINT:HTAB (9) : PRINT "R.Prust/W.VOSS 1984":PRINT
5 FOR I = 1 TO 2000 :NEXT
10 HOME:PRINT "CE PROGRAMME SERT DE DEMONSTRATION"
:PRINT:PRINT "IL EXPLIQUE DE QUELLE MANIERE ":PRINT:PRINT "INTERROMPRE ET "
PRINT:PRINT" REMETTRE EN MARCHE UN PROGRAMME"
30 GOSUB 200
40 HOME:PRINT "I",I*I":PRINT
50 FOR I = 1 TO 50
60 PRINT I, I*I : IF I/10    INT (I/10) THEN NEXT I
70 GOSUB 200 :PRINT:PRINT: IF I    50 THEN HOME:PRINT "I"
"I*I":PRINT NEXT I
80 VTAB 22:PRINT "LE PROGRAMME EST TERMINE"
90 END
200 VTAB:HTAB6: INVERSE:PRINT "APPUYEZ SUR UNE TOUCHE": NORMAL
210 POKE-16368,0:WAIT-16384,128:POKE-16368,0
220 VTAB 22:CALL-958; RETURN
```

Le programme principal ne change pas jusqu'à la ligne 900, seul le sous-programme change et encore, seulement la ligne 210.

Dans l'emplacement de mémoire 49152 (ou -16284: complément de 65536) on enregistre si une touche a été enclenchée ou si le code de la dernière touche appuyée a pu être transmis.

Pour ne pas avoir de surprises, on remet cette adresse à 0 afin que les touches appuyées auparavant ne soient pas prises en considération.
La commande WAIT est une spécialité d'APPLE, elle met en place un STOP combiné à un CONT. Le programme est interrompu jusqu'à ce que le contenu de l'adresse ne soit supérieur à la valeur précisée dans la commande. La syntaxe est la suivante

WAIT Adresse, Valeur

L'adresse de l'exemple -16384 (ou 49168) contient des informations sur la dernière entrée manuelle du clavier . Si on appuie sur une touche une valeur supérieure à 128 y est stockée.

Voyons maintenant un deuxième cas dans lequel un programme se déroule jusqu'à ce qu'END soit atteint ou bien jusqu'à ce qu'une interruption voulue par l'utilisateur ait lieu.

Nous allons de nouveau utiliser l'exemple de l'impression de nombres et de leurs carrés, sans utiliser une boucle FOR...NEXT, mais une boucle sans fin.

```
3 HOME:HTAB (9):PRINTT "APPLE TRUCS & ASTUCES"
4 PRINT:PRINT:HTAB (9):PRINT:R.PRUST /W.VOSS 1984":PRINT
5 FOR Q =1 TO 2000: NEXT
```

```
10 HOME:POKE 37,21:PRINT
20 INVERSE :PRINT "POUR TERMINER LE PROGRAMME": PRINT: HTAB 10:
PRINT"APPUYEZ SUR UNE TOUCHE";:NORMAL
30 SPEED=1500
40 POKE 35,20:HOME
50 POKE-16368,0
60 I=I+1
70 PRINT I,I*I
80 IF PEEK (-16384) 128 THEN 60
90 POKE -16368,0
100 SPEED=255:TEXT
110 END
```

Description du programme

Lignes	Contenu
10	Après l'effacement de l'écran, le curseur est placé en ligne 21, la commande sert au PRINT suivant
20	Maintenant suit un message précisant le fonctionnement du programme
30	On peut modifier la vitese d'édition avec la commande SPEED. D'habitude SPEED est égal à 255. Ici nous allons réduire cette valeur pour que les messages suivants soient plus lisibles.
40	La partie du bas de l'écran est fermée
50	On s'assure que les anciennes saisies au clavier ne soient pas prises en considération.
60-70	Calcul et sortie
80	Avec cette instruction conditionnée les lignes 60 et 70 sont exécutées jusqu'à ce qu'une valeur supérieure à 128 soit contenue à l'adresse -16384 ce qui correspond à une entrée au clavier. Dès qu'une touche est pressée

```
                la valeur à l'adresse -16384 dépasse 128, puis
                la boucle sans fin est interrompue.

90              L'adresse -16384 est remise à 0

100             La rapidité d'édition est rétablie à sa
                valeur initiale
                Avec la commande TEXT, la fenêtre d'écran
                est rétablie

110             Fin du programme
```

Les possibilités d'utilisation des interruptions de programmes seront éclaircie au chapitre 5 et dans le chapitre 7.4.

Avec ces 4 parties sur les applications de l'accès mémoire direct nous allons clore ce chapitre.

Nous aurions certainement pu continuer à développer ce problème en renonçant à quelques exemples. Mais nous préférons donner la priorité à l'utilisation pratique.

3.2 L'Assembleur

3.2.1 Bases

L'utilisateur d'un ordinateur comme l'APPLE pense souvent que le langage de programmation de base, le langage BASIC, est la seule possibilité de communication avec l'APPLE.
Mais ce n'est pas vrai. Toutes sortes de langages de programmation peuvent être utilisés par l'APPLE, si on dispose sur une disquette du programme de traduction nécessaire (compiler).
On évoquera donc différents langages, puis nous examinerons plus attentivement les particularités et les possibilités d'utilisation de l'Assembleur. Ce langage original est compris directement par l'ordinateur qui n'utilise deux chiffres: 0 et 1 qui représentent deux positions "circuit ouvert" et "circuit fermé".

Le langage machine est naturellement pour nous difficilement manipulable.
C'est pourquoi on s'est préoccupé très tôt de faire en sorte que ces éléments de langage soient des symbôles courants pour nous.

Un de ces langages est le langage Assembleur.

Cette notion d'Assembleur signifie deux choses :

1 D'une part il s'agit (comme nous l'avons déjà vu) d'un langage de programmation (orienté vers la machine).

2 D'autre part il est souvent employé avec la notion de programme de traduction, c'est à dire que le programme Assembleur est un programme

machine retraduit.

La programmation en Assembleur rend donc nécesaire un programme de traduction qui s'appelle tout simplement Assembleur et qui doit être en général enregistré sur disquette.

Le développement d'un tel langage, maintenant facilement utilisable, comporte néanmoins aussi un désavantage. La retraduction du programme d'origine (programme source) en programme machine (programme objet) prend du temps en calculs et occupe de la mémoire.

Etant donné le rendement dont est capable l'ordinateur et sa rapidité, cet inconvénient n'est cependant pas trop pesant.
Au contraire : on irait encore plus loin et on développerait des langages de programmation qui seraient utilisables encore plus facilement que les différentes versions de l'Assembleur.

Cette façon d'utiliser le langage montre qu'il n'est plus nécessaire de connaitre en détail les adresses mémoire comme avec un Assembleure. C'est pourquoi les langages qui sont surtout utilisés comme procédés de retraduction sont relativement facilement assimilables et très rapides à maîtriser.
Les langages les plus importants et les plus connus sont les suivants :

FORTRAN (Formula Translation), plus utilisé pour les problèmes techniques et scientifiques

COBOL (Common Business Oriental Langage), principalement utilisé dans le monde des affaires

ALGOL (Algorithme Langage) , plus particulièment destiné à

----- ---- - l'établissement de problèmes mathématiques

Le Basic (Beginnner's All Purpose Symbolic Instruction Code) appartient également à ce groupe

Dans ce chapitre il ne s'agit pas de faire du lecteur un parfait programmateur en Assembleur, mais de montrer les principales caractéristiques de ce langage qui permettent la compréhension de l'utilisation d'un sous-programme Assembleur dans un programme BASIC.

Pour permettre la compréhension de ces bases il faut tout d'abord avoir un aperü des deux systèmes de numération : les systèmes binaires et hexadécimal.

Le système binaire ne se rapporte qu'aux deux chiffres 0 et 1.
Maintenant pour pouvoir comprendre ce que signifie par exemple 1101101, on considère le nombre décimal 2348. Ce nombre se décompose ainsi :

2*1000+3*100+4*10+8*1
=2*10 puissance 3 +3*10 puissance 2 +4*10 puissance 1 +8*10 puissance 0

De ceci on peut déduire qu'en règle générale :
Chaque nombre peut se décomposer en une somme qui a autant de termes que le nombre décomposé a de chiffres. Chaque terme est un produit de deux facteurs, le premier facteur correspond à chaque chiffre du nombre

décomposé , le second facteur est une puissance entière de 10, les exposants des nombres entiers Z1 sont représentés en ordre décroissant , lorsque Z est le nombre de chiffres.
Ceci est valable dans le système binaire (et dans tous les autres systèmes de numérotation également) avec la différence qu'à la place de "puissance entière de 10 " on donnera des puissances de 2.
Nous pouvons maintenant expliquer le nombre 1101101 (utilisation d'un nombre binaire dans unnombre décimal).

1101101 =

1*2 +1*2 +0*2 +1*2 +1*2 +0*2 +1*2

= 1*64 +1*32 +0*16 +1*8 +1*4 +0*2 +1*1
= 64 +32 +8 +4 +1
= 109
(10)

Inversement un nombre décimal peut être converti en nombre binaire, en le "remplissant" avec une puissance entière de 2, en commençant par le plus important.
Que signifie 93 en système binaire ?

93 =
(10)

1*64 +0*32 +1*16 +1*8 +1*4 +0*2 +1*1
- - - - - - -
=1011101
(2)

Si l'on met côte à côte les nombres du système décimal et du système binaire on obtient :

Décimal	Binaire
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

On s'aperçoit que l'on a besoin de 4 chiffres en binaire lorsque l'on doit convertir les 10 chiffres du système décimal de l'ordinateur.
Pour 4 emplacements à réserver, on utilise seulement 10 symbôles différents (0,1....9) ce qui est plus facile à lire.

Décimal	Binaire
.	.
.	.
8	1000
9	1001
10	1010
11	1011
12	1100

13	1101
14	1110
15	1111

Ce tableau montre que pour 4 emplacements binaires, on ne peut placer au maximum que 16 symbôles différents (0,1....15).
Ceci explique l'importance donnée au système hexadécimal dans le traitement des données. Cela garantit une utilisatioon optimale de la mémoire que l'on a à sa disposition. On compte avec 16 chiffres dans ce système , comme suit

Déc.	Bin.	Hex.	Déc.	Bin.	Hex.
0	0	0	11	1011	B
1	1	1	12	1100	C
2	10	2	13	1101	D
3	11	3	14	1110	E
4	100	4	15	1111	F
5	101	5			
6	110	6	16	10000	10
7	111	7	17	10001	11
8	1000	8	18	10010	12
9	1001	9	19	10011	13
10	1010	A	20	10100	14
			.	.	.

Le rapport entre le système décimal et le système hexadécimal est expliqué plus simplement avec la conversion d'un système dans l'autre.
Si l'on veut convertir un nombre hexadécimal en un nombre décimal, la phase de décomposition ci-dessus est toujours valable.
Exemple :

```
        10FA    =
           (16)

    1*16 +0*16 +F*16 +A*16

   =1*4096 +0 +15*16 +10*1

   =4096  +240 +10 =4346
                      (10)
```

Si à l'inverse on veut convertir un nombre décimal en nombre hexadécimal, il faut le remplir avec la puissance 16 :

Exemple :

```
        458    =
           (10)

    1*16 +12*16 +10*16

  =1*256 +C*16 +A*1
   -      -     -
  =1CA
     (16)
```

L'observation du système hexadécimal dans ce résumé est nécessaire parce que les instructions Assembleur, que nous allons voir, sont composées de mots de 3 caractères, qui utilisent des nombres hexadécimaux. De sorte que les adresses mémoire, les données qui doivent être indiquées, sont à traiter comme nombres hexadécimaux.

Pour plus de renseignements sur l'Assembleur nous vous renvoyons aux ouvrages spécialisés (par exemple R.Mohole : Assembly language programming for the Apple II, Osborne, Mac Graw Hill, 1982).

Généralement chaque instruction Assembleur est composée de 5 parties (champs), tous les champs ne sont pas nécessairement occupés.

Instruction Assembleur, construction schèmatique :

```
-----------------------------------------------------------------
INo DE LIGNE I ETIQUETTE I MNEMONIQUE I OPERANDE I COMMENTAIRE I
I     1      I     2     I      3     I     4    I      5      I
-----------------------------------------------------------------
```

Exemple

```
1000 VAL1    EQU     30B   ; ATTRIBUTION  (LET)
1100 ADD     LDA     VAL1  ; MEMORISATION DU REGISTRE A
```

La position centrale est le champ 3 : c'est là qu'est situé le mot clé des instructions Assembleur.
Dans la ligne 1000 c'est l'instruction EQU qui indique que l'adresse mémoire

30 B (=799) est occupé par une valeur.
(16) (10)

Dans le premier champ on trouve un numéro de ligne obligatoire, au deuxième emplacement il peut y avoir une étiquette (mais pas forcément) qui peut être utilisée comme "étiquette de transfert" quand elle est utilisée comme opérande (champs 4) dans une autre instruction (voir ligne 1100).
Dans le champs 4 on trouve un opérande qui n'est valable que pour quelques instructions, comparable en quelque sorte à l'instruction INPUT en BASIC, qui ne peut se passer d'un nom de variable comme l'instruction RETURN.

On peut introduire un commentaire explicatif dans le dernier champ. Il est fréquent qu'il soit séparé par ; ou *, selon la version Assembleur

utilisée.

Si par exemple les champs de 1 à 4 ne sont pas utilisés, il apparaît des instructions de commentaires claires- qui peuvent être comparées à l'instruction REM du BASIC. Nous allons examiner maintenant un programme Assembleur complet que nous avons pris dans le livre mentionné de Mottole (pages 25/26).

Le programme suivant sert à afficher le mot ERR sur l'écran, émet deux signaux sonores sur le haut parleur de l'APPLE et empèche un saut de ligne après l'édition de 3 espaces.

Un tel programme pourrait très bien être utilisé pour indiquer , par exmple une erreur de données.

Nous reviendrons aux sous-programmes machine qui sont utilisés dans la ROM de l'APPLE. Le programme utilise principalement les adresses suivantes:

FF2D (16) : Impression de ERR et Bip
FBDD (16) : Bip
F948 (16) : impression de trois espaces

L'APPLE a déjà à sa disposition des étiquettes pour la première et la troisième adresse , respectivement PRERR et PRBLNK. Onpeut attribuer à la deuxième adresse l'étiquette BELL , c'est généralement une étiquette recommandée.)

Le programme est ainsi:

```
 1 PRBLNK EQU F948
 2 BELL   EQU FBDD
```

```
 3 PRERR  EQU FF2D
 4 ERROR  JSR PRERR  ; Impression "ERR"
 5        JSR  BELL  ; Bip
 6        JSR PRBLMK ; 3 espaces
 7        RTS        ; Return
 8        END
```

Dans les trois premières lignes les attributions nécesaires à l'étiquettage ont lieu. Dans les lignes 4-6 , on raccorde les sous-programmes qui réalisent les problèmes cités plus haut. En ligne 7 c'est le retour qui a lieu et le programme se termine avec la ligne 8.

Ce programme Assembleur ne peut rien réaliser ou pas plus qu'un programme BASIC sans commande RUN.

Avant de traiter un programme Assembleur, il faut d'abord "assembler" c'est à dire traduire d'une manière compréhensible pour la machine. Ceci se produit après l'introduction de la commande ASM (et naturellement seulement lorsque l'on a installé sur l'APPLE un programme Assembleur (traduction) suur la disquette.

Avec la commande ASM on établit à côté du programme d'origine un programme objet qui est à peu près comme cela :

```
**END OF PASS 1
**END OF PASS 2

0800           1  ;************************
0800           2  ;THIS SUBROUTINE WILL
0800           3  ;PRINT "ERR", BEEP
0800           4  ;THE SPEAKER TWICE,
0800           5  ;THEN PRINT 3 BLANKS
0800           6  ;************************
0800           7  ;
0800           8  ;
0800           9  PRBLNK   EQU  $F948
0800          10 BELL1A    EQU  $FBDD
0800          11 PRERR     EQU  $FF2D
0800          12 ;
0800          13 ;
0800 20DDFF   14 ERROR     JSR  PRERR     ;PRINT "ERR"
0803 20DDFB   15           JSR  BELL1A    ;BEEP SPEAKER
0806 2048F9   16           JSR  PRBLNK    ;PRINT 3 BLANKS
0809 60       17           RTS            ;THEN RETURN
              18           END

***** END OF ASSEMBLY
```

```
                    ************************
                    *                      *
                    * SYMBOL TABLE -- V 1.5 *
                    *                      *
                    ************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

** ABSOLUTE VARIABLES/LABELS

PRBLNK F948  BELL1A FBDD  PRERR  FF2D  ERROR  0800

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0032
```

Une fois que l'assemblage est fait, le programme généré est numéroté, en commençant en 800 (ce numéro de ligne ne change pas jusqu'à la première ligne du "noyau de programme" c'est à dire jusqu'à la ligne portant l'étiquette "ERROR".

Ceci signifie que le programme généré "proprement dit" n'est constitué que d'un groupe de lignes qui commence avec la ligne ERROR et se termine avec la ligne RETURN (=RTS).
Les lignes restantes ne sont pas assemblées car ceci n'est pas nécessaire, en efffet elles sont comparables à l'instruction BASIC REM. Il est clair que dans la liste de programme susmentionnée les lignes EQU ne sont pas assemblées non plus (ceci est comparable à l'instruction LET en BASIC).

Toutefois elles sont prises en considération. EQU ne produit donc pas d'instruction de code généré , cependant elle indique ce qu'il faut faire à l'Assembleur. Dans cette mesure, cette instruction (comme quelques autres) se situe entre une instruction qui est assemblée dans le code objet et une instruction qui est entièrement transmise (comme par exemple l'instruction commentaire qui est introduite avec ;). Une telle instruction est appelée Directive Assembleur ou "Pseudo-OP". L'instruction ORIGIN est une de ces directives particulièrement importante qui établit l'adresse de départ pour le code objet. (L'adresse de départ utilisé par l'Assembleur LISA est, par exemple, l'adresse hexadécimale 800).

Si l'on veut utiliser une autre adresse de départ, et ceci est parfois nécessaire, lorsque plusieurs programmes Assembleur doivent être extraits d'un programme principal. Ceci peut être effectué avec l'instruction ORG (=ORIGIN). Il faut ensuite que soit communiqué à partir de quelle adresse de départ le code doit être mémorisé pendant l'assemblage (et pas seulement

après).

L'instruction qui conditionne l'adresse de départ pour le programme objet s'appelle OBJ.
C'est pourquoi on écrit par exemple les deux instructions suivantes au début du programme

ORG 300 nouvelle adresse de départ du programme

OBJ 800 adresse de départ du code pendant l'assemblage

Le programme est maintenant établi de la manière suivante :

```
1  ;************************
2  ;THIS SUBROUTINE WILL
3  ;PRINT "ERR", BEEP
4  ;THE SPEAKER TWICE,
5  ;THEN PRINT 3 BLANKS
6  ;************************
7  ;
8  ;
9            ORG  $300
10           OBJ  $800
11 ;
12 ;
13 PRBLNK    EQU  $F948
14 BELL1A    EQU  $FBDD
15 PRERR     EQU  $FF2D
16 ;
17 ;
18 ERROR     JSR  PRERR      ;PRINT "ERR"
19           JSR  BELL1A     ;BEEP SPEAKER
20           JSR  PRBLNK     ;PRINT 3 BLANKS
21           RTS             ;THEN RETURN
22           END
```

Lorsque nous assemblons maintenant ce programme cela donne :

```
**END OF PASS 1
**END OF PASS 2

                                                    ************************
                                                    *                      *
                                                    * SYMBOL TABLE -- V 1.5 *
                                                    *                      *
                                                    ************************

0800          1  ;************************
0800          2  ;THIS SUBROUTINE WILL
0800          3  ;PRINT "ERR", BEEP
0800          4  ;THE SPEAKER TWICE,
0800          5  ;THEN PRINT 3 BLANKS          LABEL. LOC.  LABEL. LOC.  LABEL. LOC.
0800          6  ;************************
0800          7  ;
0800          8  ;                            ** ZERO PAGE VARIABLES:
0300          9           ORG  $300
0300          10          OBJ  $800            ** ABSOLUTE VARIABLES/LABELS
0300          11 ;
0300          12 ;                             PRBLNK F948  BELL1A FBDD  PRERR  FF2D  ERROR  0300
0300          13 PRBLNK   EQU  $F948
0300          14 BELL1A   EQU  $FBDD           SYMBOL TABLE STARTING ADDRESS:6000
0300          15 PRERR    EQU  $FF2D           SYMBOL TABLE LENGTH:0032
0300          16 ;
0300          17 ;
0300 202DFF   18 ERROR    END  PRERR      ;PRINT "ERR"
0303 20DDFB   19          JSR  BELL1A     ;BEEP SPEAKER
0306 2048F9   20          JSR  PRBLNK     ;PRINT 3 BLANKS
0309 60       21          RTS             ;THEN RETURN
              22          END

***** END OF ASSEMBLY
```

Avant de commencer ce programme on le mémorise sur une disquette pour qu'il soit disponible et que le programme ne soit pas assemblé à chaque utilisation. Le programme doit être mémorisé comme un fichier binaire, car il est codé en binaire, c'est précisemment l'avantage de l'utilisation d'un programme Assembleur (plus exactement des sous-programmes assemblés) parce que si l'on code, on peut travailler beaucoup plus vite pendant le déroulement du programme qu'avec un programme BASIC normal.
La mémorisation binaire permet de calculer la longueur du programme :

```
longueur = adresse de fin - adresse de début + 1
         =       309      -    300          + 1
         =        10     =A  (hexadécimal)
```

Les adresses de début et de fin sont prélevées de la liste d'assemblage. Il s'agit du numéro de la ligne de début et de fin du véritable code objet. Celui-ci commence comme il a déjà été expliqué dans la ligne avec l'étiquette ERROR et finit à l'instruction RTS (toutes les autres lignes ne sont pas des instructions réalisables (i) exemple les directives).

La mémorisation du programme objet utilise donc la commande suivante :

```
BSAVE FAUTE.OBJ,A800,LA
      Nom            longueur (hex)
                adresse de départ (hex)
```

Cette commande doit être introduite par l'assembleur LISA avec les touches CTRL et D.
Si ce programme doit être utilisé on a seulement besoin de la commande:

```
BLOAD FAUTE.OBJ,A300        L'adresse de début du
```

code-objet est précisée
par ORG

Comment le programme Assembleur peut être utilisé à partir du BASIC, c'est ce que nous allons développer dans le chapitre suivant. On ne pénètrera pas profondément ici les secrets de la programmation Assembleur . Ce n'est pas le but de ce livre de donner un cours d'introduction à l'Assembleur mais beaucoup plus de montrer comment utiliser les possibilités de l'ordinateur APPLE. A cet égard le contenu des prochains chapitres et les exemples déjà donnés plus hauts sont peut être un peu trop complexes. Néanmoins il n'est sûrement pas inutile de savoir au moins dans les grandes lignes comment un programme Assembleur est établi et ce qu'il faut comprendre par "assemblage".

3.2.2 Combinaison BASIC-ASSEMBLEUR

Sans acquérir son propre Assembleur, tel que le programme LISA, on peut utiliser le Moniteur de l'APPLE II pour la programmation en Assembleur .

Ce Moniteur est un programme d'aide, un utilitaire qui permet en particulier de se connecter avec un programme BASIC.

Les possibilités du Moniteur vont être développées ici rapidement, après avoir vu comment l'appeler.

Il est appelé grâce à une instruction dont nous n'avions pas beaucoup parlé jusqu'ici, l'instruction CALL:

```
nn CALL m
```

Lorsque cette instruction est appelée du BASIC, un branchement a lieu dans un sous-programme écrit en langage machine. m dans l'instruction CALL donne l'adresse du sous-programme machine.

De telles routines sont mémorisées en mémoire morte (donc non effaçable) entre:

D000 et FFFF (hexadécimal)
ou 53248 et 65535 (décimal)

La première instruction du Moniteur se trouve à l'adresse FF69(=65385 déc.). Avec CALL, m est en décimal. Toutes les adresses supérieures à 32767 (déc.) sont notées négativement.

Cette adresse négative est égale à l'adresse positive moins 65536.

L'adresse de départ du Moniteur est 65385 soit

65385-65536=-151

Pour appeler le Moniteur nous avons donc besoin de l'instruction BASIC suivante:

CALL -151

Le curseur sous APPLESOFT (]) sera maintenant * afin d'indiquer à l'utilisateur qu'il est sous contrôle du Moniteur.

Pour revenir au BASIC il suffit d'utiliser les touches:

CTRL-C

puis la touche RETURN.

Le curseur] APPLESOFT apparaît de nouveau.

Que faire avec le Moniteur après l'avoir appelé avec CALL -151 ?

Tout d'abord cet utilitaire peut être utilisé pour lire le contenu d'un octet:

On entre alors:

```
CALL -151
```

et après l'apparition de l'étoile on entre l'adresse, par exemple, 11FF, le Moniteur répond:

```
11FF- 24
```

Ce qui signifie que dans l'emplacement mémoire 11FF on trouve la valeur hexa. 24 soit 36 décimal.

Si maintenant on actionne la touche RETURN, l'ordinateur donne le contenu des emplacements mémoire qui suivent 11FF.

Sur chaque ligne, après l'appui de la touche RETURN, on compte 8 contenus de mémoire et un peu moins si l'adresse n'est pas divisible par huit.

Si on entre:

```
00FF
```

L'ordinateur affiche:

```
OOFF- 10
```

et après l'appui sur RETURN huit valeurs apparaissent:

```
0100- 31 35 31 00 30 30 30 30
```

car 0100 (hexa.)/256 (déc.) est divisible par huit.

Si on veut inspecter en même temps différents emplacements mémoire, on peut entrer deux adresses séparées par des points.

Tous les contenus de mémoire entre ces deux adresses sont affichés.
Exemple

```
F800.F83F
```

Il est possible aussi de modifier ces contenus:
Si on entre par exemple:

```
FF
```

Le moniteur affiche:

```
OOFF- 10
```

Si on tape:

```
:09
```

à l'emplacement 00FF on a la valeur 09, ce qu'on peut vérifier en entrant:

```
FF
```

Le moniteur affiche:

```
00FF- 09
```

Le moniteur à d'autres possibilités que d'inspecter la mémoire ou de la transformer (Aucun nombre supérieur à FF (hexa.)/ 255 (déc.) ne pourra être entré dans un emplacement).
Les autres possibilités vont être décrites rapidement:

Afin de faciliter le travail il existe une instruction permettant le transfert de zones mémoire.

Après l'appel du Moniteur si on entre l'instruction:

```
1200 2000.2100M
```

Les contenus des emplacements 2000 à 2100 sont transférés en 1200.
Le transfert des informations se fera bien plus rapidement qu'avec l'instruction LET du BASIC dans une boucle FOR...NEXT.

Il va de soi que l'effacement de zones mémoire peut ainsi être effectué plus vite qu'avec LET.
Pour cela simplement on a juste besoin de l'adresse A qui précède la zone à effacer et à laquelle on attribue la valeur 0 puis le contenu de A sera copié de A+1 à A+X.

Ainsi on copie le 0 de A dans A+1, le 0 de A+1 dans A+2 etc...

Une instruction du Moniteur permet de comparer les contenus de mémoire.

A la place de M, on utilise V.

Le Moniteur ne donne pas de concordance lors de la comparaison.

Une telle procédure de comparaison peut être très utile pour vérifier si un fichier a été correctement transféré sur une disquette ou si des erreurs de transmissions ont eu lieu.
Cette opération s'appelle Vérification.

Cette vérification peut se dérouler ainsi:
Avec un fichier de données constitué de 100 octets en mémoire centrale (de 2000 à 2064 hexa.) que l'on désire mémoriser, il faut retourner au BASIC avec:

CTRL-C et RETURN

Puis, mémoriser cette zone avec:

BSAVE DONNEES,A2000,L64

BSAVE permet la mémorisation en fichier binaire au lieu du SAVE habituel. On précisera l'adresse de début de la zone ainsi que sa longueur en hexadécimal.

Si la zone à partir de 2100 est libre on peut entrer:

BLOAD DONNEES,2100

On peut alors comparer avec le Moniteur:

```
CALL -151
2000 2100.2164V
```

Si le Moniteur ne renvoie pas d'erreur, la mémorisation est réussie, sinon il faut recommencer l'opération.

Le Moniteur possède des possibilités de branchement.
Pour cela on donne une adresse suivie de la lettre G.

Exemple:

```
CALL -151
3D0G
```

Cette instruction permet un branchement en 3D0, adresse à laquelle se trouve l'instruction machine:

```
JMP 9DB9
```

JMP signifie branchement (JUMP) et l'adresse 9DB9 permet de reconnaître le programme machine d'une disquette avec le programme de compilation Applesoft.

L'instruction G n'est rien d'autre qu'un JMP.

Il existe un programme en assembleur dans le Moniteur.
Il faut néanmoins préciser qu'il s'agit d'un assembleur qui ne permet la résolution d'un nombre restreint de problèmes.

L'adresse de départ de cet Assembleur est F666 (hex.)=-2458 (déc.).On l'appelle donc par:

```
CALL -151
F666G
```

ou directement par:

```
CALL -2458
```

Cet assembleur ne peut être utilisé que si l'APPLE II possède la carte INTEGER BASIC ou s'il s'agit d'un APPLE standard.

Lorsque l'assembleur est prêt il se signale d'un point d'exclamation (au lieu de pour INTEGER-BASIC ou * pour le Moniteur).

Les adresses seront alors précédées du signe $.

Si un emplacement mémoire doit être inspecté on tapera après !:

```
$1CFF
```

L'Assembleur affiche:

```
1CFF -6E
```

On retourne au Moniteur avec FF69G, précédé de $ donc:

```
$FF69G
```

On retourne à l'APPLESOFT avec:

```
$ CTRL-C
```

Les instructions possibles avec ce mini-assembleur ne seront pas développées car comme nous l'avons déjà expliqué cet assembleur n'est pas disponible sur toutes les versions APPLE.
Pour cela nous vous renvoyons à la littérature spécialisée.

3.3 Sous-programmes en langage machine

Il existe un certain nombre de routines en langage machine qu'on peut utiliser comme nous allons le voir par la suite.

L'utilisateur n'a pas besoin d'écrire ces sous-programmes, il suffit de les appeler par l'instruction CALL dont nous avons déjà parlé.
On pourra utiliser ces routines pour développer un programme.

Les plus importantes concernent les problèmes suivants:

1. Représentation graphique
2. Tester une touche
3. Edition de texte
4. Edition d'un signal sonore
5. Déplacement du curseur
6. Présentation de l'écran

Pour illustrer ceci, voici quelques exemples courts:

Exemple 1: Edition d'un "bip"

Le programme suivant génére quelques "bip" lors de l'utilisation d'un sous-programe appelé par:

```
CALL -1052
```

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT: PRINT TAB(9)"R.PRUST / W. VOSS 1984": PRINT
```

```
50 PRINT : PRINT: PRINT TAB(5)"3.11- SOUS PROGRAMME BIP"
60 GOSUB 1000: REM ATTENDRE
70 FOR I=1 TO 15
80 PRINT I;".FOIS:BIP"
90 CALL -1052
100 FOR J=1 TO 99: NEXT J
110 NEXT I
120 PRINT: PRINT: PRINT"    DE L'EDITION": END
1000 REM SP ATTENDRE
1010 VTAB23: HTAB6: INVERSE: PRINT"SVP APPUYEZ SUR UNE TOUCHE!": NORMAL
1020 GET A$: IF A$="" THEN 1020
1030 HOME : RETURN
```

Description du programme

Lignes	Contenu
30-60	Titre et attendre
70-110	Le son "bip" est émis 15 fois (avec un affichage) 100: entre les sons chaque fois de courtes bouclese d'attente
120	Fin de programme
1000-1030	Sous-programme d'attente 1010: message 1020: attendre l'appui d'une touche 1030: effacement de l'écran et retour au programme principal

Exemple 2: Déplacement du curseur

Le programme suivant vous montre, comment le curseur peut-être déplacé vers le haut. Pour cela nous utilisons un sous programme qui est appelé avec:

```
CALL -998
```

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT: PRINT TAB(9)"R.PRUST / W. VOSS 1984"
```

```
50 PRINT : PRINT: PRINT TAB(7)"3.12-DEPLACEMENT DU CURSEUR"
60 GOSUB 1000: REM ATTENDRE
70 FOR I=1 TO 5
80 PRINT I;".EDITION"
90 NEXT I
100 PRINT: PRINT: PRINT"IMPRESSION DE 5 ESPACES"
110 PRINT: PRINT: PRINT: PRINT:PRINT: PRINT"LES VOILA"
120 PRINT: PRINT"APRES CONT LE CURSEUR VA ETRE DEPLACE"
130 PRINT "DE 12 LIGNES VERS LE HAUT,A L'ENDROIT OU ** AHA **"
135 PRINT"A ETE IMPRIME": PRINT:PRINT"TAPER CONT": STOP
140 FOR I=1 TO 12
150 CALL -998
160 NEXT I
165 FLASH
170 PRINT TAB(15)"** AHA **           "
175 NORMAL
180 VTAB 23: PRINT"FIN DE L'EDITION": END
1000 REM SP ATTENDRE
1010 VTAB 23: HTAB 6: INVERSE: PRINT" SVP APPUYER SUR UNE TOUCHE!": NORMAL
1020 GET A$: IF A$="" THEN 1020
1030 HOME: RETURN
```

Description du programme

Lignes	Contenu
30-60	Titre et attendre (SP 1000)
70-90	5 impressions du mot sortie
100	Indication de 5 lignes vides à suivre.
110	Impression de 5 lignes vides
120-135	Explication du programme et interruption
140-160	Déplacement du curseur 12 lignes vers le haut
165-175	Edition de ** AHA ** en FLASH.(Les particularités de FLASH et NORMAL seront évoquées au chapitre 4.)
180	Fin du programme après le message correspondant en ligne écran 23
1000-1030	Sous-programme attente 1010 Message en ligne 23 colonne 6 en inverse puis retour en mode normal. 1020 Attente de l'appui d'une touche 1030 Retour

Exemple 3: Messages d'erreur

Le programme montre l'utilisation d'une routine en langage machine appelée avec CALL -211 et qui permet d'envoyer un signal sonore et un message en cas d'erreur:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT: PRINT TAB(9)"R.PRUST / W. VOSS 1984"
50 PRINT : PRINT: PRINT TAB(10)"3.13-ANNONCE D'ERREUR"
60 GOSUB 1000: REM ATTENDRE
70 HOME
80 PRINT"ANNONCE D'UNE ERREUR : ": PRINT: PRINT: PRINT
90 CALL -211
95 PRINT: PRINT: PRINT: PRINT: PRINT
100 PRINT: PRINT: PRINT"FIN": END
1000 REM SP ATTENDRE
1010 VTAB 23: HTAB 6: INVERSE: PRINT" SVP APPUYER SUR UNE TOUCHE!": NORMAL
1020 GET A$: IF A$="" THEN 1020
1030 HOME: RETURN
```

Description du programme:

Lignes	Contenus
30-60	Titre et attente (SP 1000)
70-80	Effacement de l'écran et édition d'un message explicatif.
90	Appel du sous-programme qui émet ERR (=erreur) et d'un Bip.
95-100	Fin de programme
1000-1030	Sous-programme d'attente (voir détails exemple 2)

3.4 Exemples d'utilisation

Dans de nombreux programmes on utilise un sous-programme permettant d'interrompre le déroulement d'un programme jusqu'à ce que l'utilisateur appuie sur une touche quelconque.

Ce sous-programme est à peu près comme suit:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT: PRINT: PRINT TAB(9)"R.PRUST / W.VOSS 1984"
50 PRINT: PRINT: PRINT TAB(8)"3.14 - ATTENTE/V1"
60 PRINT: PRINT: PRINT : PRINT: PRINT"CE PROGRAMME ATTEND L'APPUI SUR UNE
TOUCHE"
70 PRINT"AVEC BRANCHEMENT DANS LE SOUS-PROGRAMME 1000"
80 PRINT
90 GOSUB 1000: REM ATTENDRE
100 PRINT"J'AI ATTENDU": PRINT: PRINT
110 PRINT"JUSQU'A CE QU'UNE TOUCHE SOIT APPUYEE"
120 PRINT"LE PROGRAMME EST TERMINE": END
1000 REM SP ATTENDRE
1010 VTAB 23:HTAB 6: INVERSE
1020 PRINT"SVP APPUYEZ SUR UNE TOUCHE"
1030 NORMAL
1040 GET A$: IF A$="" THEN 1040
1050 HOME : RETURN
```

Voici ce qui se passe après le REM de la ligne 1000.

1. En 1010 une impression en inverse est préparée colonne 6 (HTAB 6)/ligne 23 (VTAB 23). INVERSE permet l'édition noir sur blanc.

2. Cette édition sort en 1020

3. En 1030 on revient au mode normal (blanc sur noir avec NORMAL)

4. En ligne 1040 on attend une entrée au clavier. Si A$="" on continue à attendre avec le branchement en 1040.

5. Si une touche est appuyée on continue en 1050. L'écran est alors effacé (HOME) et on retourne au programme principal (RETURN)

Dans quelques uns des exemples précédents nous avons regroupés ces 10 instructions en moins de 6 lignes.

Un tel programme d'attente, peut être établi avec l'aide d'un sous-programme machine perfectionné. Il se déroulera alors plus vite et on lui évitera l'instruction GET, qui combinée avec du graphisme haute résolution (cf. Chapitre 4), occasionne de graves erreurs lors des sorties graphiques.

Programmation alternative du programme d'attente

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT: PRINT: PRINT TAB(9)"R.PRUST / W.VOSS 1984"
50 PRINT: PRINT: PRINT TAB(8)"3.15 - ATTENTE/V2"
60 PRINT: PRINT: PRINT : PRINT: PRINT"CE PROGRAMME ATTEND L'APPUI SUR UNE
TOUCHE"
70 PRINT"AVEC BRANCHEMENT DANS LE SOUS-PROGRAMME 1000"
80 PRINT
90 GOSUB 1000: REM ATTENDRE
```

```
100 PRINT"J'AI ATTENDU": PRINT: PRINT
110 PRINT"JUSQU'A CE QU'UNE TOUCHE SOIT APPUYEE"
120 PRINT"LE PROGRAMME EST TERMINE": END
1000 REM SP ATTENDRE
1010 VTAB 23
1020 CALL -868
1030 INVERSE: HTAB 6
1040 PRINT"SVP APPUYEZ SUR UNE TOUCHE"
1050 NORMAL
1060 POKE -16368,0
1070 WAIT -16384,128
1080 POKE -16368,0
1090 VTAB 23
1100 CALL -868
1110 RETURN
```

Ce programme peut être écrit différemment, on peut le raccourcir de deux lignes:

Ligne 1000: Instruction 1000
Ligne 1010: Instructions 1010-1110

Nous décrirons le programme plus haut pour profiter de sa numérotation plus pratique:

Description du programme:

Lignes	Contenu
1000	Commentaire (REM)
1010	Positionnement ligne 23
1020	Effacement de la ligne par la routine machine
1030	Mode INVERSE et positionnement colonne 6
1040	Impression du texte
1050	Mode NORMAL
1060	Adresse -16368 remise à 0
1070	Attente d'une entrée au clavier (Code ASCII =128)
1080	Adresse -16368 remise à 0 après l'entrée clavier, afin que cette entrée ne soit pas utilisée.
1090-1100	Effacement de la ligne 23
1100	Retour

Dans l'exemple suivant, nous allons utiliser les possibilités de déplacement du curseur vues dans la partie précédente.

Dans de nombreux programmes, l'utilisateur est invité à répondre à une question posée par l'ordinateur avec O (oui) ou N (non). Il faut vérifier si l'utilisateur a utilisé un autre symbôle que O ou N, si c'est le cas, un Bip le signale et le curseur reste en place attendant une entrée correcte (O ou N).
Le programme continue ensuite.

Sans cette vérification le programme ressemble à celui-là:

```
100 HOME
110 PRINT"DOIS-JE CONTINUER (O/N) ?"
115 PRINT: PRINT: PRINT
120 INPUT"ENTREZ O OU N: ";A$
130 IF A$="N" THEN 110
140 PRINT: PRINT: PRINT"ON CONTINUE"
150 REM SUITE DU PROGRAMME
160 PRINT: PRINT: PRINT"FIN": END
```

Ce programme réagit à l'entrée correcte de O. Si on appuie sur une mauvaise touche cette entrée sera traitée comme un oui.
Le programme suivant fonctionne mieux:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT: PRINT: PRINT TAB(9)"R.PRUST / W.VOSS 1984"
50 PRINT: PRINT: PRINT TAB(3)"3.16 - CORRECTION D'ENTREES ERRONEES"
60 GOSUB 1000: REM ATTENDRE
70 HOME
80 PRINT"LE PROGRAMME DOIT-IL CONTINUER ?"
90 PRINT: PRINT: PRINT
100 PRINT"SVP ENTREZ O OU N";:GET A$: PRINT: IF A$ "O" AND A$ "N" THEN
CALL -1052: CALL -998 :GOTO 100
120 IF A$="N" THEN PRINT: PRINT: PRINT: GOTO 80
130 PRINT: PRINT: PRINT : PRINT"LE PROGRAMME PEUT MAINTENANT CONTINUER"
140 REM SUITE DU PROGRAMME
150 PRINT: PRINT: PRINT: PRINT"FIN": END
1000 REM SP ATTENDRE
1010 VTAB 23: HTAB 6: INVERSE: PRINT"SVP APPUYEZ SUR UNE TOUCHE": NORMAL
1020 GET A$: IF A$="" THEN 1020
1030 HOME : RETURN
```

Voici la description du programme:

Lignes	Contenu
30-50	Titre
60	Attente (SP 1000)
70	Effacement de l'écran
80-90	Message
100	Sauts de ligne
110	On doit entrer O ou N Saisie avec GET Saut de ligne Si A$ est différent de O ou N (entrée erronée), émission d'un Bip (CALL -1052) et positionnement du curseur sur la ligne précédente, et rebranchement en ligne 100
120	Si N est entré la question de la ligne 80 est répétée Il serait possible de traiter différemment la réponse
130-140	Le programme continue
150	Fin du programme
1000-1030	Sous-programme d'attente (déjà expliqué à d'autres endroits)

Dans un dernier exemple nous allons montrer de quelle manière on peut corriger des énoncés erronnés:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT:PRINT:PRINT TAB(9)"R. PRUST/W. VOSS 1984"
50 PRINT:PRINT: PRINT TAB(5)"3.18-CORRECTIONDES DONNEES FAUSSES"
60 GOSUB 1000:REM ATTENDRE
70 HOME
80 PRINT"SVP DONNEZ LIGNE SUIVANTE"
90 INPUT"UN NOMBRE DECIMAL:";X
100 PRINT: PRINT: PRINT
110 GOSUB 3000: REM CORRECTION
120 PRINT:PRINT: PRINT"MAINTENANT LE PROGRAMME PEUT CONTINUER"
130 REM SUITE DU TRAITEMENT DES DONNEES
140 PRINT:PRINT:PRINT"FIN":END
150 PRINT: PRINT: PRINT: PRINT"FIN":END
1000 REM SP ATTENDRE
1010 VTAB 23: HTAB 6:INVERSE: PRINT"SVP APPUYEZ SUR UNE TOUCHE!": NORMAL
1020 GET A$: IF A$="" THEN 1020
1030 HOME: RETURN
3000 REM SP CORRECTION
3010 GOSUB 6000: REM VERIFICATION
3020 IF A$="O" THEN RETURN
3030 PRINT: PRINT"ENONCE FAUX:";X
3040 PRINT:PRINT: INPUT "CORRECTION     :";X
3050 GOTO 3010
6000 REM SP VERIFICATION
6010 PRINT"EST-CE QUE L'ENTREE EST CORRECTE?(O/N)";:GET A$:PRINT:IF A$  "O"
AND A$  "N" THEN CALL -1052: CALL -998: GOTO 6010
6020 CALL -998: CALL -868: RETURN
```

Description du programme:

Lignes	Contenu
30-50	Titre
60	Attente avec branchement dans le sous-programme 1000
70	Effacement de l'écran
80-90	Demande d'un nombre décimal pour les tests
100	Lignes vides
110	Branchement dansle sous-programme 3000,pour vérifier le chiffre donné (80-90) et éventuellement le corriger
120-130	Après le branchement le programme peut continuer
140	Fin du programme
1000-1030	Sous-programme d'attente (déjà expliqué ailleurs)
3000-3050	Sous-programme pour la correction de valeurs erronées 3010: Branchement dans le sous-programme 6000 à vérifier 3020: Avec l'entrée "0" (pour oui) dans le sous-programme 6000,la preuve de la correction du test est faite et suit un retour au programme principal. 3030: Si le chiffre testé est erroné,il est écarté 3040: Ensuite le chiffre corrigé est demandé 3050: Rebranchement en 3010 pour vérifier également cette nouvelle entrée(3040)
6000-6020	Sous-programme pour la vérification des valeurs

données:

6010: On demande si la valeur donnée est correcte
Vérification de l'entrée "O" (pour oui),ou "N" (pour non)

6020: Création de l'emplacement de sortie et retour dans le sous-programme 3000 (ligne 3020).

Ces exemples devraient suffire à montrer comment il est possible d'améliorer le dialogue avec l'ordinateur. L'utilisation du programme machine apporterait encore bien des améliorations.

Chapitre 4: Programmes graphiques

4.1 Remarques préliminaires

L'établissement de graphiques avec l'ordinateur est un des problèmes les plus intérressants et les plus importants.
Comme tous les ordinateurs ont des exigences graphiques différentes, l'utilisateur pourra avec un choix approprié les différencier les uns des autres. Certains des ordinateurs ont des possibilités graphiques, d'autres non. Certains le deviennent plus ou moins, avec l'aide de coûteux logiciels, d'autres les possèdent déjà.

L'APPLE appartient à cette deuxième catégorie
Il faut prêter attention au fait que l'APPLE et quelques autres ordinateurs ont des possibilités graphiques différentes:

1.Les graphiques normaux
2.Les graphiques à haute résolution

Dans le premier cas il faut penser que l'écran est divisé en 40 colonnes et 48 lignes et que chaque "point de rencontre" correspondant peut être utilisable séparemment.

Des 48 lignes qui existent en mode graphique seules 44 lignes (0 jusqu'à 43) peuvent être utilisées, le reste est réservé à deux lignes de texte en bas de l'écran.

Chaque point dans ce mode graphique est relativement "grossier"(sur un écran de 30 cm environ 3*5mm)

Ceci est différent avec des graphiques en haute résolution:

Il faut alors penser à partager l'écran en 192 lignes et 80 colonnes, de manière à ce qu'apparaissent environ 53760 "points de rencontre" utilisables séparemment. Ceci rend possible l'élaboration de graphiques beaucoup plus fins et même de graphiques haute résolution.

4.2 Graphique normal

Le graphique normal est une possibilité d'établissement graphique qui n'est pas très intéressante à utiliser en BASIC. Il est plutôt propre à l'établissement de questions spécifiques que nous montrerons dans des exemples ultérieurs.

4.2.1 Modes texte et graphique

Normalement l'écran se trouve en mode texte, c'est à dire que lorsqu'un programme est tapé, les lignes du programme apparaissent, comme texte lisible à l'écran (par chance!). Ensuite apparaissent les résultats donnés par l'instruction PRINT.
Si par exemple maintenant on crée des graphiques, il faut d'abord commuter l'écran en mode graphique.

Cette commutation s'établit avec l'instruction:

GR (GR="graphique")

Cette instruction peut être utilisée comme commande que comme instruction à l'intérieur d'un programme en BASIC (naturellement pourvu de lignes numérotées)

Lorsque le programme atteind cette instruction, il y a commutation en mode graphique. C'est à dire que le texte existant sur l'écran (par exemple le listing) iisparait. De toute manière les deux lignes du bas de l'écran sont laissées libres pour un texte.

Lorsque dans la suite du programme les instructions appropriées au graphique suivent, l'image correspondante apparaît sur l'écran.

Souvent on retourne à la fin de l'édition du graphique au mode texte. Soit parce que par exemple des sorties en texte suivent, soit parce qu'il faut corriger une erreur quelquonque ou simplement parce que le programme doit être continué ou allongé.

Ce retour en mode texte est opéré avec l'instruction:

```
TEXT
```

Cette instruction peut être aussi utilisée comme une instruction BASIC.

Si cela arrive pendant le traitement du programme, le graphique disparaît. C'est pourquoi il est préfèrable d'utiliser avant cette instruction,l'instruction STOP ou GET pour donner à l'utilisateur la possibilité de regarder son graphique tranquillement.

Ceci est formulé à peu près comme cela:

```
SVP entrez CONT     (avec STOP) ou:
SVP appuyez sur une touche   (avec GET)
```

Avec ceci, celui qui a peu ou pas d'expérience en programmation en BASIC sait ce qu'il faut faire.

Ce texte peut être imprimé sur l'une des deux lignes du bas de l'écran, sans avoir dû auparavant utiliser l'instruction TEXT. Car ces lignes sont à disposition dans chaque cas pour la sortie d'un texte.

Pour écrire à la colonne 23 on utilise:

```
VTAB 23        (VTAB=tabulation verticale)
```

4.2.2 Couleurs et effets graphiques

Il existe des possibilités indépendantes du mode graphique pour produire des effets graphiques: les instructions

INVERSE

ou

FLASH

L'instruction INVERSE transforme une sortie de texte "blanc sur noir" en sortie de texte "noir sur blanc".

On arrive avec l'instruction FLASH à ce que l'impression des deux possibilités soit interchangeable en permanence.

On peut supprimer ces impressions particulières avec l'instruction:

NORMAL

Voici le programme suivant:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT: PRINT: PRINT TAB(9)"R. PRUST/W. VOSS 1984"
50 PRINT: PRINT TAB(10)"4.1-EFFETS SPECIAUX"
60 PRINT: PRINT: PRINT: PRINT: PRINT
70 INVERSE: PRINT"APPLE":NORMAL:PRINT:PRINT
80 FLASH: PRINT"ECLAIR": NORMAL
90 PRINT: PRINT: PRINT"FIN":END
```

L'APPLE permet encore plus:
S'il est équipé d'une interface couleur, d'un téléviseur couleur ou d'un

moniteur couleur, on peut également produire des sorties couleur.

Pour cela on utilise l'instruction:

```
COLOR = valeur
```

Les valeurs situées entre 0 et 15 établissent la couleur de l'édition selon le tableau suivant:

Tableau des couleurs

couleur	valeur
noir	0
magenta	1
bleu marine	2
pourpre	3
vert foncé	4
gris 1	5
bleu moyen	6
bleu ciel	7
brun	8
orange	9
gris 2	10
rose	11
vert clair	12
jaune	13
bleu outremer	14
blanc	15

4.2.3 Instructions graphiques

Les instructions qui sont introduites en mode graphique simple vont être expliquées rapidement avant d'être illustrées par des exemples.

La première instruction s'appelle:

PLOT S,Z

Elle sert à dessiner le point de rencontre de la colonne S avec la ligne Z (en forme de petit rectangle)

Nous rappelons qu'il y a 40 lignes à disposition (elles sont désignées de 0 à 39) et 44 lignes pour le graphique (0 à 43).C'est pourquoi S ne peut être ni négatif ni supérieur à 39 et Z ne peut être ni négatif ni supérieur à 43.

Si l'on ne suit pas cette règle, une faute est annoncée de la manière suivante:

ILLEGAL QUANTITY ERROR IN nn

Les valeurs S et Z peuvent être calculées (exemple: résultat d'un calcul ou variable), à condition qu'elles soient contenues dans les limites décrites.

Avec cette instruction on peut aussi dessiner des traits sur l'écran, pour cela il suffit d'inclure l'instruction PLOT dans une boucle comme le montre l'exemple suivant:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT: PRINT: PRINT TAB(9)"R. PRUST/W. VOSS 1984"
50 PRINT: PRINT TAB(7)"4.2-LIGNE VERTICALE"
```

```
60 GOSUB 1000: REM ATTENDRE
70 HOME
80 GR:COLOR=12:REM VERT
90 FOR I=5 TO 20
100 PLOT 10,1
110 NEXT I
120 GOSUB 1000: REM ATTENDRE
130 TEXT: HOME : PRINT"FIN":END
1000 REM SP ATTENDRE
1010 VTAB 23: HTAB 6:INVERSE:PRINT"SVP APPUYEZ SUR UNE TOUCHE!"
1020 NORMAL
1030 GET A$: IF A$= "" THEN 1030
1040 RETURN
```

```
VLIN Z1,Z2 AT S
```

Cette instruction dessine une ligne verticale dans la colonne S de la ligne Z1 àla ligne Z2
De la même manière

```
HLIN S1,S2 AT Z
```

dessine une ligne horizontale dans la ligne Z qui commence à la colonne S1 et finit à la colonne S2.

Il faut conserver les limites pour les valeurs S,S1,S2,Z,Z1 et Z2.

4.2.4 Exemple 1: Ligne d'écran

Pour l'utilisation de nombreux programmes il faut dessiner une bordure à l'écran, ceci peut se faire en suivant le programme suivant:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT: PRINT: PRINT TAB(9)"R. PRUST/W. VOSS 1984"
50 PRINT: PRINT TAB(14)"4.3-LIGNE"
60 GOSUB 1000: REM ATTENDRE
70 HOME
80 INPUT"COLONNE A GAUCHE    :";S2
90 IF S1 0 OR S1 39 THEN PRINT: PRINT"ENTREE FAUSSE": PRINT: GOTO 80
100 PRINT: INPUT"COLONNE DE DROITE   :";S2
110 IF S2 0 OR S2 39 THEN PRINT: PRINT"ENTREE FAUSSE":PRINT: GOTO 100
120 PRINT: INPUT"LIGNE DU HAUT   :";Z1
130 IF Z1 0 OR Z1 43 THEN PRINT:PRINT"ENTREE FAUSSE":PRINT: GOTO 120
140 PRINT: INPUT"LIGNE DU BAS   :";Z2
150 IF Z2 0 OR Z2 43 THEN PRINT: PRINT"ENTREE FAUSSE":PRINT:GOTO 140
160 GR: COLOR=12: REM VERT
170 HLIN S1,S2 AT Z1
180 HLIN S1,S2 AT Z2
190 VLIN Z1,Z2 AT S1
200 VLIN Z1,Z2 AT S2
210 GOSUB 1000: REM ATTENDRE
220 TEXT: HOME : PRINT"FIN":END
1000 REM SP ATTENDRE
1010 VTAB 23: HTAB 6:INVERSE:PRINT"SVP APPUYEZ SUR UNE TOUCHE!"
1020 NORMAL
1030 GET A$: IF A$= "" THEN 1030
1040 RETURN
```

4.2.5 Exemple 2: Grille

Comme exercice nous allons introduire un petit programme, qui dessine sur l'écran une grille, sans possibilité de vérifier les erreur. On laisse à l'utilisateur, l'appréciation de l'espacement horizontal ou vertical.

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT: PRINT: PRINT TAB(9)"R. PRUST/W. VOSS 1984"
50 PRINT: PRINT TAB(13)"4.4-GRILLE"
60 GOSUB 1000: REM ATTENDRE
70 HOME
80 INPUT"ECART HORIZONTAL  :";W
90 PRINT: INPUT"ECART VERTICAL  :";S
100 GR: COLOR=12: REM VERT
110 FOR I=0 TO 39 STEP S
120 VLIN 0,39 AT I: NEXT I
130 FOR I=0 TO 39 STEP W
140 HLIN 0,39 AT I: NEXT I
150 GOSUB 1000: REM ATTENDRE
160 TEXT: HOME : PRINT"FIN":END
1000 REM SP ATTENDRE
1010 VTAB 23: HTAB 6:INVERSE:PRINT"SVP APPUYEZ SUR UNE TOUCHE!"
1020 NORMAL
1030 GET A$: IF A$= "" THEN 1030
1040 RETURN
```

4.2.6 Exemple 3: Démonstration de couleur

Dans le programme suivant les possibilités de couleur vont être expliquées de façon simple. Naturellememt ce programme n'a de sens que si l'on possède la couleur PAL de l'APPLE (interface couleur) et un téléviseur ou un moniteur couleur.

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT: PRINT: PRINT TAB(9)"R. PRUST/W. VOSS 1984"
50 PRINT: PRINT TAB(13)"4.5-COULEURS"
60 GOSUB 1000: REM ATTENDRE
70 GR
80 FOR F=0 TO 15
90 COLOR=F
100HLIN 5,35 AT 2*F
120NEXT F
130 GOSUB 1000: REM ATTENDRE
140 TEXT: HOME : PRINT"FIN":END
1000 REM SP ATTENDRE
1010 VTAB 23: HTAB 6:INVERSE:PRINT"SVP APPUYEZ SUR UNE TOUCHE!"
1020 NORMAL
1030 GET A$: IF A$= "" THEN 1030
1040 RETURN
```

Lorsque l'on n'a pas la carte couleur on peut utiliser des trames de différentes sortes.

4.2.7 Exemple 4: Histogramme

Un histogramme est l'image graphique de la fréquence d'une répartition statistique.

Pour illustrer ce type de programme on utilise l'exemple suivant:

Répartition des tranches d'âge en République Fédérale Allemande en 1980:

Tranches d'âge	Nombre	%
- 10	6317.0	10.3
10 à 20	10087.7	16.4
20 à 30	8964.2	14.6
30 à 40	8263.1	13.4
40 à 50	8714.0	14.2
50 à 60	7332.7	11.9
60 à 70	5489.7	8.9
70 à 80	4800.3	7.8
80 à 90	1469.2	2.4
+ 90	128.5	0,2
Somme	61566.3	100.1 *

Source:Statistique de l'annuaire de la République Fédérale Allemande 1982 p.59.
* nombre arrondi

Dans la dernière colonne du tableau précédent, les fréquences relatives des données seront établies en graphique avec des barres de longueurs

différentes.

Voici le programme BASIC que l'on peut établir à partir de cela:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT :PRINT : PRINT TAB(9)"R.PRUST / W. VOSS 1984"
50 PRINT: PRINT TAB(11)"4.6- HISTOGRAMME"
60 GOSUB 1000: REM ATTENDRE
70 N=10: REM NOMBRE DE CLASSES
80 IF N 39 THEN PRINT"TROP DE CLASSES. REDUISEZ LE NOMBRE DE CLASSES": GOTO
500
90 DIM F(N), FT(N)
100 FOR I=1 TO N: READ F(I) :NEXT I
110 VALEUR MAX. F
120 FM=0
130 FOR I=1 TO N: IF F(I) FM THEN FM=F(I)
140 NEXT I
150 REM STANDARDISATION
160 FOR I=1 TO N: FT(I)=F(I)*(35/FM)
170 FT(I)=INT(FT(I)*100+.5)/100: NEXT I
180 SW=3
200 REM DESSIN
210 GR: COLOR=12: REM VERT
220 HLIN 0,39 AT 35
230 FOR I=1 TO N
240 VLIN35-FT(I),35 AT I*SW+5
250 NEXT I
260 VTAB 21: PRINT TAB(8)"10 20 30 40 50 60 70 80 90  90"
270 GOSUB 1000: REM ATTENDRE
490 TEXT: HOME
500 PRINT"FIN EN 500": END
600 DATA 10.3,16.4,14.6,13.4,14.2,11.9,8.9,7.8,2.4,0.2
1000 REM SP ATTENDRE
```

```
1010 VTAB 23: HTAB 6:INVERSE: PRINT"SVP APPUYEZ SUR UNE TOUCHE"
1020 NORMAL
1030 GETA$:IF A$="" THEN 1030
1040 RETURN
```

Explication rapide du programme:

Lignes	Contenu
30-50	Titre
60	Attente
70-80	Nombre de classes, si 39 le nombre de classes doit être réduit pour permettre un affichage correct.
90	Dimensionnement
100	Lecture des données
110-140	Détermination de la fréquence maximale
150-170	Les fréquences sont calculées de manière à ce que la fréquence maximale couvre 35 lignes.
180	Pas de progression. Si plus de 12 classes SW=2, si plus de 17, SW=1 (voir aussi 70-80)
200-210	Préparation du dessin
220	Tracé d'une ligne
230-250	Dessin des barres de l'histogramme
260	Impression d'un axe horizontal, à modifier selon les programmes
270	Attente
490	Mode texte et impression finale
1000-1040	Sous-programme d'attente

4.2.8 Exemple 5: Jeu de balle

Des éléments graphiques issus des exemples précédents on peut développer facilement un modèle de base pour les jeux de mouvements. Ceci sera le dernier exemple de ce groupe.

Il s'agit de la version la plus simple d'un jeu de balle dans lequel une balle représentée par un rectangle se déplace entre les bordures de l'espace de jeu (cf. 4.2.4).

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT :PRINT : PRINT TAB(9)"R.PRUST / W. VOSS 1984"
50 PRINT: PRINT TAB(11)"4.7- JEU DE BALLE"
60 GOSUB 1000: REM ATTENDRE
70 GR
80 VTAB 23: HTAB 6: PRINT"ATTENDEZ 12 SEC. SVP"
90 COLOR=12: REM VERT
100 REM CADRE
110 VLIN 0,39 AT 0: VLIN 0,39 AT 39
120 HLIN 0,39 AT 0: HLIN 0,39 AT 39
130 REM POSITION DE DEPART
140 X=10: Y=25
150 REM PAS DE PROGRESSION ET NOUVELLES COORDONNEES
155 XV=1: YV=1
160 NX=X+XV: NY=Y+YV
170 IF NX 38 THEN NX=38: XV=-XV
180 IF NX 1 THEN NX=1: XV=-XV
190 IF NY 38 THEN NY=38: YV=-YV
200 IF NY 1 THEN NY=1: YV=-YV
210 COLOR=12: PLOT NX,NY: REM BALLE VERTE
220 COLOR=0: PLOT X,Y: REM EFFACEMENT DE L'ANCIENNE POSITION
230 X=NX: Y=NY
```

```
240 I=I+1: IF I =250 THEN 160
250 VTAB 24: INPUT"ON RECOMMENCE (O/N) ?";A$
260 IF A$="O" THEN I=0 :GOTO 160
270 TEXT: HOME: PRINT"FIN": END
1000 SP ATTENDRE
1010 VTAB 23: HTAB 6:INVERSE: PRINT"SVP APPUYEZ SUR UNE TOUCHE": NORMAL
1020 GET A$: IF A$="" THEN 1020
1030 RETURN
```

Lignes	Contenu
30-60	Remarques préliminaires et attente
70	Commutation en mode graphique
80	Effacement de la ligne du bas
90	Impression du cadre coloré
100-120	Délimitation du champ de jeu
130-140	Position de départ (Elle peut être modifiée).
150-155	Pas de progression. Il détermine la vitesse de la balle et la direction.
160	Nouvelles coordonnées de la balle
170-200	Si le bord du champ est atteint la balle revient On peut ici déterminer l'angle de rebond et faire varier la vitesse.
210-220	Une balle verte est dessinée à la nouvelle position; une balle noire efface l'ancienne position, de manière à ce que l'illusion optique établisse le mouvement.
230	La nouvelle position est considérée comme ancienne et sera effacée au déplacement suivant
240	Le processus se répète jusqu'à ce que I soit égal à 250.
250-260	Le jeu peut continuer le programme recommence en 160
270	Impression de fin
1000-1030	SP attendre

Si on veut introduire une deuxième balle dans le champ de jeu (Il faudra alors donner d'autres coordonnées de départ et un nouveau pas de progression) et comparer les coordonées des deux balles on pourra vérifier si elles se touchent et interrompre le programme avec un BOUM.

Le lecteur cherchera à établir lui-même un tel programme.

4.3 Graphiques haute résolution

Nous avons déjà expliqué que pour un graphique haute résolution il faut penser à partager l'écran en 192 lignes et 280 colonnes. Chacun de ces 53760 points sont adressables séparement d'où la possibilité avec APPLE de dessiner des lignes très fines. Ce qui est appréciable dans le domaine des mathématiques comme nous le verrons plus loin.
Comme dans le graphisme normal, les lignes du bas de l'écran seront laissées libres de manière à ce qu'il reste effectivement 160 lignes et 280 colonnes.

Il reste un point à observer lors de l'utilisation d'un graphique haute résolution:
Le besoin en mémoire est considérable et il arrive fréquemment qu'un programme trop long vienne perturber l'édition du graphique (et en général le détruise)
On peut néanmoins résoudre ce problème:
Lorsque ces effets apparaissent (ils sont le plus souvent reconnaissables, l'ordinateur dispersant à la place du graphisme des points irréguliers), ils peuvent être résolus avec les instructions HIMEM et LOMEM.
Deux pages écrans sont réservées dans l'APPLE pour l'édition de deux graphiques.Si on utilise la page I ,on fixera LOMEM à 16384.
Ainsi le programme mémorisé ne sera pas perturbé par le graphique. On relèvera d'autres particularités dans les manuels pour utilisateur.
Si le programme n'est pas trop important on ne tiendra pas compte de ce qui a été dit plus haut.

4.3.1 Préparation d'un graphique haute résolution

Le graphisme haute résolution est introduit par l'instruction

HGR

Elle efface la première page graphique et prépare l'édition du graphisme haute résolution (HGR2 correspond à la page 2)
L'écran est composé, comme nous l'avons déjà vu, de 280 colonnes et 160 lignes, il reste 4 lignes de texte en bas de l'écran.
Il n'y a pas de lignes de texte pour la page 2, on entre donc: HGR2: POKE -16301,0.(Les textes seront mémorisés par POKE, cette possibilité n'intervient que dans des cas exceptionnels et ne sera pas développée ici).

La commutation en mode texte se fait par:

TEXT

4.3.2 Couleurs

Il est possible d'utiliser six couleurs différentes (avec, évidemment, une carte et un moniteur couleur).
Voici l'instruction correspondante:

HCOLOR=Valeur

La valeur est comprise entre 0 et 7, voici les couleurs correspondantes:

Couleurs:

Valeur	Couleur	Valeur	Couleur
0	Noir	4	Noir
1	Vert	5	Orange
2	Violet	6	Bleu
3	Blanc	7	Blanc

4.3.3 Les instructions graphiques

De même que pour le graphisme normal, il existe une instruction permettant de dessiner un point ou une ligne.
Pour dessiner un point on utilisera l'instruction:

```
HPLOT S,Z
```

Cette instruction dessine un point à la colonne S, ligne Z

L'instruction

```
HPLOT S1,Z1 TO S2,Z2
```

dessine une ligne entre les points P1(S1,Z1) et P2(S2,Z2)
En élargissant cette instruction on peut dessiner une suite de lignes comme le montre l'exemple suivant:

```
HPLOT 10,10 TO 279,159 TO 50,50
```

4.3.4 Exemple 1: Système de coordonnées

Dans ce premier exemple nous allons montrer comment placer un système de coordonnées en ligne Z colonne S (avec S compris entre 0 et 279 et Z entre 0 et 159)

ATTENTION! Comme dans d'autres programmes, il n'y a aucune vérification des données entrées. Le lecteur pourra établir ces vérifications par la suite.

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST / W.VOSS 1984"
50 PRINT: PRINT TAB(11)"4.8- AXE DES COORDONNEES"
60 VTAB 20:PRINT"SVP TAPEZ CONT": STOP
70 HOME
80 INPUT"COLONNE DE DEPART";S
90 INPUT"LIGNE DE DEPART";Z
100 HGR: HCOLOR=7: REM BLANC
110 HPLOT 0,Z TO 279,Z
120 HPLOT S,0 TO S,159
130 VTAB 23:PRINT"SVP TAPEZ CONT": STOP
140 TEXT: HOME : PRINT"FIN": END
```

On fera attention avec ce programme et les suivants à la couleur choisie si on travaille avec un moniteur N/B.

4.3.5 Exemple 2: Fonction linéaire

Si une droite est dessinée dans un système de coordonnées on positionnera l'origine à la colonne S=0 et à la ligne Z=159
Le programme précédent peut être repris avec ces nouvelles données.

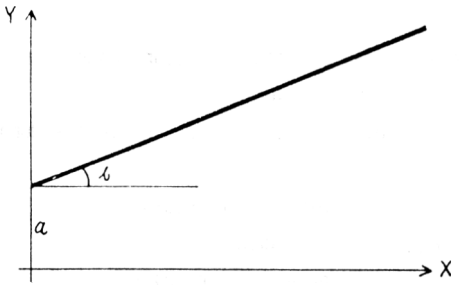
La droite est déterminée par les deux paramètres:

-Segment de l'ordonnée a

-Pente (tangente de l'angle) b

conformément à la fonction:

$$Y_i = a + bX_i$$



Le programme permet d'entrer les paramètres a et b dans deux INPUT. On peut alors dessiner sans problème des droites et des systèmes de cordonnées.

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT : PRINT TAB(9)"R.PRUST / W.VOSS 1984"
50 PRINT: PRINT TAB(13)"4.8- DROITES"
60 VTAB 20:PRINT"SVP TAPEZ CONT": STOP
70 HOME : S=0: Z=159
80 INPUT"A= ";A
90 PRINT: INPUT"B= ";B
100 HGR: HCOLOR 7: REM BLANC
110 HPLOT 0,Z TO 279,Z
120 HPLOT S,0 TO S,159
130 FOR X=0 TO 279
140 Y=159-A-B*X
150 IF Y 0 OR Y 159 THEN 170
160 HPLOT X,Y
170 NEXT X
180 VTAB 23: PRINT"FIN": END
```

Description du programme:

Lignes	Contenu
30-50	Titre
60	Interruption du programme
70	Effacement de l'écran et détermination des coordonnées du système de coordonnées
80-90	Paramètres de la droite
100	Commutation en mode haute résolution
110-120	Dessin des axes
130-170	Dessin de la droite
	130 Détermination de X
	140 Détermination de Y en fonction de X
	150 Vérification de Y
	160 Dessin d'un point de la droite
	170 Point suivant
180	Impression finale et fin du programme

4.3.6 Exemple 3: Le cercle

L'exemple simple du cercle montre qu'il est possible de tracer des courbes en haute résolution.
Les instructions INPUT permettent d'établir le centre et le rayon du cercle.
Le dessin peut être établi point par point, chaque point étant défini en fonction de Xi (=coordonnée de colonne) et Yi (=coordonnée de ligne)
On utilisera l'équation du cercle suivante pour un centre quelconque:

$$(X_i - X_o) + (Y_i - Y_o) = Y$$

dans laquelle:

$(Y)^{1/2}_i$ = coordonnée Y

X_i = coordonnée X

R= rayon

X_o = coordonnée x du centre

Y_o = coordonnée y du centre

Les valeurs de X se déplaçant dans un domaine compris entre R-Xo et R+Xo, les valeurs (Y1)i constituent la partie supérieure du cercle et (Y2)i la partie inférieure.
On veillera à ce que les valeurs de X et Y ne dépassent pas les limites

déterminées.

Voici le programme:

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT: PRINT TAB(9)"R.PRUST / W.VOSS 1984"
50 PRINT: PRINT: PRINT: PRINT TAB(13)"4.10- CERCLE"
60 VTAB 20: PRINT"SVP TAPEZ CONT": STOP
70 HOME: S=0: Z=159
80 INPUT" COORDONNEE X DU CENTRE: ";A
90 INPUT" COORDONNEE Y DU CENTRE: ";B
95 PRINT: PRINT: INPUT"RAYON: ";R
100 HGR: HCOLOR=7: REM BLANC
110 HPLOT 0,Z TO 279,Z
120 HPLOT S,0 TO S,159
130 FOR X=A-R TO A+R
140 H=R*R-(X-A) 2
150 IF H =0 THEN 200
160 Y1=B+SQR(H): Y2=B-SQR(H)
170 IF X 0 OR X 279 OR Y1 0 OR Y1 159 THEN 190
180 HPLOT X,Y1
190 IF X 0 OR X 279 OR Y2 0 OR Y2 159 THEN 200
195 HPLOT X,Y2
200 NEXT X
210 VTAB 23:PRINT"FIN": END
```

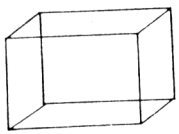
La description du programme est moins développée que les précédentes car plusieurs parties ont déjà été commentées.

Lignes	Contenu
30-50	Titre
60	Interruption
70	Effacement et détermination du système de coordonnées
80-95	Détermination du centre et du rayon
100	Commutation en mode haute résolution
110-120	Dessin des axes
130-200	Dessin du cercle
	130 Détermination de X
	140 Détermination de H
	150 Elimination des valeurs de H non recevables
	160 Détermination de Y pour les moitiés sup. et inf. du cercle
	170 Elimination des valeurs non recevables
	180 Dessin de la moitié inférieure
	190 Elimination des valeurs non recevables
	195 Dessin de la moitié supérieure
	200 Points suivants
210	Fin du programme

4.3.7 Exemple 4: Graphique en trois dimensions

Il est naturellement possible d'établir des graphiques en trois dimensions, cela demande un peu plus de peine.

Voici un cube:



et le programme correspondant avec des données fixes pour la longueur, la largeur, la profondeur, et les angles.(Il va de soi que ces données pourraient être entrées avec des INPUT).

```
30 PRINT TAB(9)"APPLE TRUCS & ASTUCES"
40 PRINT : PRINT: PRINT TAB(9)"R.PRUST / W.VOSS 1984"
50 PRINT: PRINT: PRINT: PRINT TAB(13)"4.11- CUBE"
60 VTAB 20: PRINT"SVP TAPEZ CONT": STOP
70 HOME
80 X1=50: X2=100: X3=70: X4=120
90 Y1=50: Y2=80 : Y3=30: Y4=60
```

```
100 HGR: HCOLOR=7: REM BLANC
110 HPLOT X1,Y1 TO X2,Y1 TO X2,Y2 TO X1,Y2 TO X1,Y1
130 HPLOT X3,Y3 TO X4,Y3 TO X4,Y4 TO X3,Y4 TO X3,Y3
140 HPLOT X1,Y1 TO X3,Y3
150 HPLOT X2,Y1 TO X4,Y3
160 HPLOT X2,Y2 TO X4,Y4
165 HPLOT X1,Y2 TO X3,Y4
170 VTAB 23: PRINT"FIN": END
```

On pourrait continuer ce programme de manière à établir une image mouvante en trois dimensions. Un tel concept est utilisé en architecture. Il existe, à ce titre, une série de logiciels et d'utilitaires qui évitent de se donner de la peine inutilement.

4.4 Le mélange de texte et de graphique

Il arrive que des graphiques et des textes doivent être mélangés à l'écran.

Il est parfois souhaitable que l'axe des coordonnées ainsi que la graduation de ces axes apparaissent en même temps que le graphique lui-même.

Pour cela il existe différentes possibilités qui ne sont pas toutes satisfaisantes:

1.Des parties de texte peuvent être placées dans les lignes du bas de l'écran comme dans les exemples précédents.

2.Dans un graphique les chiffres ou les caractères à placer peuvent être "dessinés" dans la mesure où un caractère peut être traité comme une partie du graphique.

3.Il existe aussi des programmes d'aide appropriés qui permettent le mélange de textes et de graphiques.

Il n'y a rien à ajouter au point 1 mais nous allons voir en détail les deux suivants.

4.4.1 Dessiner un caractère.

Si l'on examine un caractère sur l'écran par exemple le 1, on s'aperçoit que ce chiffre est composé d'une rangée de points et pourrait être dessiné à l'intérieur d'un écran commuté en mode graphique avec une série d'instructions HPLOT.
Ce procédé a trois avantages:

1.Les caractères apparaissent sur le graphique à l'écran (c'est le but de cette méthode.)
2.Ces caractères peuvent être placés à des emplacements quelconques de l'écran
3.Enfin ils peuvent avoir des tailles différentes.
Le programme suivant en est l'un des fondements:

```
30 PRINT TAB(9)"APPLE TRUCS & ATUCES"
40 PRINT : PRINT :PRINT
50 PRINT TAB(8)"R,PRUST / W.VOSS 1984"
60 PRINT : PRINT : PRINT : PRINT TAB(9)"4.13 DESSIN DU CHIFFRE 1"
70 PRINT : PRINT : PRINT"CE PROGRAMME DESSINE LE CHIFFRE 1"
80 PRINT"AVEC DIFFERENTES TAILLES"
90 PRINT"ET DIFFERENTES EPAISSEURS DE TRAIT"
95 PRINT"A N'IMPORTE QUELLE POSITION DE L'ECRAN"
96 PRINT : PRINT"ATTENTION : " : PRINT "CE PROGRAMME NE VERIFIE PAS LES
VALEURS" : PRINT"NON VALIDES"
97 GOSUB 5000 : REM ATTENTE
120 GOSUB 1000 : REM 1
130 PRINT : PRINT"FIN": END
1000 REM SP 1
1010 HOME : INPUT"FACTEUR D'AGRANDISSEMENT: ";F%
1015 PRINT : INPUT"EPAISSEUR DU TRAIT: ";ST
1020 PRINT : INPUT"COORDONNEES (S,Z) : ";S,Z
```

```
1025 HGR : HCOLOR=3
1040 FORK=1 TO ST
1050 HPLOT S+F%*2,Z TO S+F%*3,Z
1070 HPLOT S+F%*3,Z TO S+F%*3,Z+F%*8
1080 HPLOT S+F%,Z+F%*8 TO S+F%*5,Z+F%*8
1090 HPLOT S,Z+F%*2 TO S+F%*2,Z
1110 S=S+1 : Z=Z+1
1120 NEXT K
1130 GOSUB 5000: REM ATTENTE
1140 TEXT : RETURN
5000 REM SP ATTENTE
5010 VTAB 23 : CALL -868 : INVERSE : HTAB 6 : PRINT"APPUYEZ SUR UNE  TOUCHE
SVP. " : NORMAL : POKE -19368,0: WAIT -16384,128 : POKE 16368,0 : VTAB 23 :
CALL -868 : RETURN
```

Ce programme dessine le chiffre 1 à un endroit déterminé, la taille et l'épaisseur du trait sont paramétrables. L'emplacement à occuper est fixé par la colonne et la ligne du coin en haut à gauche. On peut aussi développer d'autres programmes semblables à celui-ci pour d'autres chiffres, lettres ou caractères particuliers qui seront gardés en réserve comme "matériau" et utilisés en cas de nécessité.

Voici une rapide description de programme pour expliquer ces fonctions.

Lignes	Contenu
30-96	Titre et explications
97	Branchement dans sp 5000: attendre
120	Branchement dans sp 1000: pour dessiner
130	Fin du programme
1010-1015	Introduction de facteurs d'agrandissement et d'épaisseur

```
          du trait du 1
1020      Choix des coordonnées de départ (S= colonne,
          Z=ligne)
          (Il s'agit du coin en haut à gauche de la matrice dans
          laquelle le 1 sera dessiné, comme le montre le dessin
          ci-contre...)

1025      Commutation en mode graphique et détermination de la couleur
1040-1120 Dessin du chiffre 1
          1040 Répétition en fonction de l'épaisseur du trait ST
          1050-1090 Dessin
          1110 Déplacement selon l'épaisseur du trait
-------------------------------------------------------------------
1130      Branchement dans sp 5000: attendre
1140      Commutation en mode texte et retour
-------------------------------------------------------------------
5000      Début de sp attendre
5010      Edition d'un message en écriture inverse, "APPUYEZ SUR UNE
          TOUCHE SVP" et retour
-------------------------------------------------------------------
```

On remarque dans cette description qu'avec le déplacement des lignes et des colonnes, le dessin du chiffre 1 dans 1050-1090, est répété aussi souvent que l'épaisseur du trait le demande. Les lignes 1050 à 1090 constituent le "noyau" du programme.Si un autre symbole que le 1 doit être dessiné, on placera d'autres instructions HPLOT à cet endroit.

4.4.2. Programme d'aide pour le mélange de textes et de graphiques.

Le mélange de textes et de graphiques pose des problèmes pour la plupart des ordinateurs, en particulier lorsqu'il s'agit de graphiques à haute définition. Lors de la commutation en mode graphique, sur la première des deux pages écran, les deux lignes du bas sont encore disponibles pour le texte, mais non dans la deuxième page écran.

Ces deux lignes ne peuvent pas être utilisées pour tout: si par exemple on désire placer une lettre ou un chiffre au milieu d'un graphique, pour la description d'un axe de système de coordonnées, ou pour le repérage de séries ou de fonctions mathématiques.

Ce problème peut être résolu avec un programme d'aide graphique approprié,mais nous n'en parlerons pas dans ce livre. Le lecteur intéressé trouvera les informations nécessaires dans n'importe quel bon logiciel.

Pour finir on peut aussi établir des caractères spéciaux, en saisissant la suite des vecteurs le constituant, chercher leur code binaire et ensuite si besoin est réutiliser cette table de formes.

Cette méthode est compliquée et nous ne la développerons pas plus car on en trouve une description claire dans le manuel de Lon Poole Martin, Mc Niff et Steven Cook (Apple II manuel d'utilisation Munich 1981 pages: 6- 13).

Chapitre 5: Amélioration des structures d'un programme

5.1 Remarques préliminaires

Quelques unes des commandes que l'on rencontre dans un programme vont être expliquées après une courte introduction, avec des exemples de programmes. Les commandes de branchements GOTO et IF...THEN ne sont pas les seules instructions qui permettent un branchement à l'intérieur d'un programme. La commande GOSUB a déjà été utilisée plusieurs fois, mais d'importantes remarques faisaient défaut. Ceci va être précisé maintenant. L'utilisateur, devra faire un compromis lors de l'élaboration d'un programme. Les deux objectifs "vitesse d'exécution maximale" et "occupation mémoire minimale" ne sont généralement pas réalisés en même temps. Ensuite d'autres objectifs viennent s'ajouter: un écran bien disposé, le confort d'utilisation, la bonne lisibilité du programme, les possibilités de transformation; ils rendent tout compromis difficile. Chaque utilisateur choisira celui des aspects cités qui lui parait être le plus important et en associera peut être d'autres à sa réflexion.

Celui qui ne programme que pour lui-même n'a sûrement pas autant d'exigences qu'un simple utilisateur; si il écrit des programmes pour des amis ou des connaissances, (qui ne sont peut être pas exercés à l'utilisation d'un ordinateur), il sera tenté d'améliorer son travail.

5.2. Les sous-programmes

On peut programmer sans sous-programme de manière satisfaisante. L'abandon d'un sous-programme et donc d'un branchement permet une meilleure compréhension du listing, des méthodes de travail, et du déroulement du programme. Mais ce style de programmation comporte aussi des désavantages. Si une même suite d'instructions doit être utilisée plusieurs fois, selon les circonstances le programme sera plus long donc moins lisible dans son ensemble et nécessitera plus de mémoire. La frappe d'un tel programme sera plus longue que nécessaire. Il est donc difficile de se passer complètement de sous-programmes.

Les sous-programmes sont une suite d'instructions qui pour la plupart n'ont pas recours aux commandes d'entrées des données; les valeurs des variables viennent souvent du programme principal. Il y a généralement au moins un ensemble de commandes qui accomplissent des problèmes clairement délimités. Si on prend comme exemple le programme du chapitre 4: dans le programme principal les valeurs nécessaires pour le graphique sont calculées, le sous-programme permet le dessin. De nombreux sous-programmes peuvent être écrits (la seule limite étant l'occupation mémoire). Les sous-programmes peuvent appeler de leur côté d'autres sous-programmes. Lorsqu'une suite de commandes apparaît plusieurs fois dans un programme, il est plus intéressant de les grouper en sous-programme c'est à dire en bloc indépendant qui sera placé au début ou à la fin du programme principal; ce bloc étant appelé avec la commande GOSUB. Souvent les sous-programmes courts, comme par exemple la boucle d'attente conditionnelle avec GET ou WAIT, sont écrits eu début de programme; par contre les sous-programmes plus longs sont plutôt placés derrière la commande END du programme principal. Ceci est lié à la vitesse de traitement.
Chaque branchement à l'intérieur d'un programme demande, selon l'instruction de branchement utilisée, un temps qui est lié à l'éloignement du branchement. A chaque instruction de branchement, l'ordinateur cherche à

partir du début du programme la ligne et l'adresse du branchement voulu. Un sous-programme avec de petits numéros de lignes sera trouvé plus vite que celui dont les numéros de lignes sont plus grands. Pour parvenir à une relation raisonnable entre le temps necéssaire à l'exécution du sous-programme et celui nécessaire au branchement, il faut respecter la règle citée plus haut. Dans tous les cas les sous-programmes situés au début doivent être sautés avec GOTO de manière à ce qu'ils ne soient pas exécutés dès le début du programme. La construction schématique est à peu près la suivante:

1. GOTO- branchement sur le début du programme.

2. Sous-programme court, qui est appelé par le programme principal ou d'autres sous-programmes.

3. Début du programme principal: initialisation avec dimensionnement, définition des fonctions, établissement des constantes et lecture éventuelle des données.
Noyau du programme.

4. END- Fin du programme principal.

5. Long sous-programme.

Un des autres inconvénients des sous-programmes pour l'APPLE réside dans la structure du langage BASIC. Les valeurs des variables que le sous-programme prend dans le programme principal, doivent être attribuées dans le programme principal à des noms de variables, qui seront également utilisées dans le sous-programme. Ceci est plus clair avec d'autres langages. Deux listes de variables peuvent être établies l'une en face de l'autre, l'une pour les variables du programme principal, l'autre pour celle du sous-programme; la transmission des valeurs est fonction de la position dans le listing.

Maintenant pour un exemple d'utilisation nous en revenons encore une fois au programme de classement du chapitre 1, mais de manière un peu différente:

Les deux tableaux de chaînes sont d'abord lus (N$ pour les noms et A$ pour les tranches d'âge), puis, lors d'une première phase le programme trie les noms et affiche le nom et l'âge correspondant ,lors de la deuxième phase le tri a lieu en fonction des âges. L'impression comprend le titre et cinq éléments triés. Pendant le processus de classement un message précise la phase en cours. Le programme existe en trois versions: une sans sous-programme, une utilisant les sous-programmes de façon excessive et une intermédiaire.

Nous commençons avec un exemple à ne pas suivre. Le programme principal ne permet rien d'autre qu'une initialisation, ce qui signifie ici dimensionnement, établissement des constantes, lecture des données, transfert de variables du programme principal au sous-programme et enfin comme dernière impression "FIN".
Tout le reste est contenu dans les sous-programmes. Le lecteur comprendra que ce programme est difficilement lisible surtout en ce qui concerne l'affichage. L'occupation mémoire y est plus importante que pour les deux autres versions, comme nous le verrons plus tard.

EXEMPLE A NE PAS SUIVRE

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 REM          **** EXEMPLE DE DEMONSTRATION ****
11 REM    ** POUR LA TECHNIQUE DES SOUS-PROGRAMMES **
20 GOTO 200
30 VTAB22: CALL -868: HTAB6: INVERSE: PRINT"APPUYEZ SUR  UNE  TOUCHE  SVP":
NORMAL: POKE-16368,0: WAIT :POKE -16384,128: POKE  -16368,0:  VTAB22  :CALL
-868: RETURN
40 H$=Y$(I): Y$(I)=Y$(J): Y$(J)=H$: H$=X$(I): X$(I)=X$(J): X$(J)=H$: RETURN
50  FOR Q=1 TO 40: PRINT"*";: NEXT: PRINT: RETURN
60 HOME: GOSUB 50: HTAB6:PRINT"PROGRAMME DE TRI": PRINT: GOSUB 50: RETURN
70 Z=0: FOR I=1 TO N: PRINT N$(I),A$(I): Z=Z+1: IF Z=M AND I N  THEN  GOSUB
90:Z=0
80 NEXT I :GOSUB 90 : RETURN
90 PRINT : GOSUB 50: GOSUB 30: GOSUB 60: RETURN
100 FOR I=1 TO N-1: FOR J=I+1 TO N: IF Y$(I) Y$(J) THEN GOSUB 40
110 NEXT J,I: RETURN
120 GOSUB 60: VTAB12: HTAB 15: INVERSE: PRINT R;".PHASE": NORMAL:  FOR  Q=1
TO 1000: NEXT: RETURN
200 REM DEBUT DU PROGRAMME
210 N=20
220 HOME: TEXT : DIM N$(N),A$(N),Y$(N),X$(N)
230 M=5
240 FOR I=1 TO N: READ N$(I),A$(I): NEXT
250 R=1: GOSUB 120: FOR I=1 TO N: Y$(I)=N$(I): X$(I)=A$(I): NEXT: GOSUB 100
260 FOR I=1 TO N: N$(I)=Y$(I): A$(I)=X$(I): NEXT: GOSUB 60: GOSUB 70
270 R=2: GOSUB 120: FOR I=1 TO N: Y$(I)=A$(I): X$(I)=N$(I): NEXT :GOSUB 100
280 FOR I=1 TO N: A$(I)=Y$(I): N$(I)=X$(I): NEXT : GOSUB 60: GOSUB 70
290 VTAB12: HTAB15: INVERSE: PRINT"FIN": NORMAL: END
1000 DATA PIERRE,40,PAUL,27,JACQUES,36,LUCIEN,56
```

```
1010 DATA RENE,24,MICHEL,45,CATHERINE,34,SIMONE,28
1020 DATA FRANCOIS,55,LEON,48,LOUIS,76,CLAUDE,40
1030 DATA LUC,25,ANNE,35,MURIELLE,34,SERGE,55
1040 DATA MARTINE,44,JEROME,52,SYLVIE,22,LEOPOLD,32
```

Nous allons chercher, par la description du programme, à rendre le déroulement un peu plus clair.

Lignes	Description
10-11	Remarques
20	Branchement sur le début du programme
30	Sous-programme d'interruption
40	Sous-programme de modification si un élément est mal placé.
50	Sous-programme pour l'impression d'une ligne d'étoiles
60	Sous-programme pour l'impression du titre;la ligne d'étoiles sert d'encadrement.
70-90	Sous-programme d'impression. Le nombre de lignes à afficher est compté par Z; après 5 lignes (M=5), et après la dernière impression le sous-programme saute à la ligne 90.Ceci permet l'impression du trait, l'attente et la nouvelle impression du titre par le branchement dans chacun des sous-programmes.
100-110	Sous-programme pour la comparaison des variables de tri Y$
120	Sous-programme pour l'impression de la phase. On utilise le sous-programme 60 pour établir le nouveau titre.
200	Début du programme principal.
210-230	Initialisation -4 tableaux doivent être dimensionnés pour pour que les sous-routines 40 et 100 puissent être utilisées
240	Lecture des données.
250	On transmet la valeur 1 à R pour le sous-programme 120,et le message est imprimé. Ensuite les tableaux X$ et Y$ sont chargés avec les valeurs de N$ et A$. Y$ est trié, X$ transmis. Le sous-programme 100 permet la comparaison à l'intérieur du tableau Y$ et on appelle si besoin est le sous-programme 40.

260 La liste triée doit être transmise aux tableaux de départ. Ensuite le sous-programme 60 est de nouveau appelé pour le le titre, le sous-programme 70 qui permet l'édition de la liste qui appelle de son côté le sous-programme 90, qui lui-même appelle les sous-programmes 50, 30 et 60 lequel appelle de nouveau le sous-programme 50- (est-ce clair ?)

270 De la même manière que dans 250, les valeurs de R doivent être transmises au sous-programme 120 et les valeurs de A$ et N$ dans le sous-programme 100 pour que le tri puisse être effectué (avec l'aide du sous-programme 40)

280 Correspond à la ligne 260: les valeurs triées doivent être à nouveau transmises au programme principal et l'impression a lieu.

290 Le programme se termine avec l'impression de "FIN"

1000-

1040 Lignes de DATA pour les données.

Lorsqu'un programme est construit de cette manière,la méthode de travail n'est pas compréhensible sans la description du programme. Au plus tard après le quatrième niveau de GOSUB on ne sait plus très bien à quelle ligne l'appel s'est produit lorsque on n'est pas l'auteur du programme. La répétition de suites d'instructions par de nombreux sous-programme doit être évitée bien que certaines commandes soient nécessaires notamment pour la transmission de valeurs de variables. De plus cette programmation exige le dimensionnement de deux tableaux supplémentaires. Si N est supérieur à 20, un gaspillage de mémoire se fait sentir.

La deuxième version utilise les sous-programmes 30 (attente) ,40 (échange), 50 (trait) et 70 (impression). Tous les autres sont supprimés, les instructions correspondantes sont déplacées. Si 90 est également déplacé l'impression de la sous-routine doit être transformée. Elle est désormais

la seule à appeler d'autres sous-programmes. Les lignes modifiées sont commentées et comparées avec la version précédente. Le sous-programme 60 (245 dans la nouvelle version) pourrait être écrit d'une façon plus intéressante (avec l'aide d'une fenêtre):

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 REM          **** EXEMPLE DE DEMONSTRATION ****
11 REM    ** POUR LA TECHNIQUE DES SOUS-PROGRAMMES **
20 GOTO 200
30 VTAB22: CALL -868: HTAB6: INVERSE: PRINT"APPUYEZ SUR  UNE  TOUCHE  SVP":
NORMAL: POKE-16368,0: WAIT :POKE -16384,128: POKE  -16368,0:  VTAB22  :CALL
-868: RETURN
40 H$=N$(I): N$(I)=N$(J): N$(J)=H$: H$=A$(I): A$(I)=A$(J): A$(J)=H$: RETURN
50  FOR Q=1 TO 40: PRINT"*";: NEXT: PRINT: RETURN
70 HOME: Z=0: FOR I=1 TO N: PRINT N$(I),A$(I): Z=Z+1: IF Z=M AND  I=N  THEN
Z=0: PRINT: GOSUB 50: GOSUB 30 : REM  TRAIT ET ATTENDRE
80 NEXT I :RETURN
200 REM DEBUT DU PROGRAMME
210 N=20
220 HOME: TEXT : DIM N$(N),A$(N)
230 M=5
240 FOR I=1 TO N: READ N$(I),A$(I): NEXT
245 GOSUB 50: HTAB6: PRINT"PROGRAMME DE TRI":PRINT: GOSUB 50:POKE 34,6:HOME
250 VTAB12: HTAB17: INVERSE : PRINT "1.PHASE": NORMAL
253 FOR I=1  TO  N-1:FOR  J=I+1  TO  N:  IF  N$(I) N$(J)THEN  GOSUB  40:REM
PERMUTATION
256 NEXT J,I
260 GOSUB 70: REM IMPRESSION DE LA LISTE TRIEE
270 VTAB12: HTAB17: INVERSE : PRINT"2.PHASE": NORMAL
273 FOR I=1  TO  N-1:FOR  J=I+1  TO  N:  IF  A$(I) A$(J)THEN  GOSUB  40:REM
PERMUTATION
276 NEXT J,I
280 GOSUB 70: REM IMPRESSION DE LA LISTE TRIEE
290 VTAB12: HTAB15: INVERSE: PRINT"FIN": NORMAL: END
1000 DATA PIERRE,40,PAUL,27,JACQUES,36,LUCIEN,56
1010 DATA RENE,24,MICHEL,45,CATHERINE,34,SIMONE,28
1020 DATA FRANCOIS,55,LEON,48,LOUIS,76,CLAUDE,40
```

```
1030 DATA LUC,25,ANNE,35,MURIELLE,34,SERGE,55
1040 DATA MARTINE,44,JEROME,52,SYLVIE,22,LEOPOLD,32
```

Ici nous n'avons pas besoin de description du programme. Après la première version celle-ci est compréhensible. Il s'agit d'une version "modérée" qui ne nécessite pas de transferts particuliers de variables dans les sous-routines.

La troisième version est mise en place sans sous-programme et elle n'est pas la meilleure:

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 REM          **** EXEMPLE DE DEMONSTRATION ****
11 REM    ** POUR LA TECHNIQUE DES SOUS-PROGRAMMES **
200 REM DEBUT DU PROGRAMME
210 N=20
220 HOME: TEXT : DIM N$(N),A$(N)
230 M=5
240 FOR I=1 TO N: READ N$(I),A$(I): NEXT
245 FOR Q=1 TO 40: PRINT"*";: NEXT: PRINT: HTAB6: PRINT"PROGRAMME DE
TRI":PRINT:FOR Q=1 TO 40: PRINT"*";: NEXT: PRINT: POKE 34,6: HOME
250 VTAB12: HTAB17: INVERSE: PRINT"1.PHASE": NORMAL
253 FOR I=1 TO N-1:FOR J=I+1 TO N:IF N$(I) N$(J) THEN H$=N$(I):
N$(I)=N$(J): N$(J)=H$: H$=A$(I): A$(I)=A$(J): A$(J)=H$
256 NEXT J,I
260 HOME: Z=0: FOR I=1 TO N: PRINT N$(I),A$(I): Z=Z+1
263 IF Z=M OR I=N THEN Z=0: PRINT :FOR Q=1 TO 40: PRINT"*";: NEXT:
PRINT:VTAB22:CALL-868: HTAB6: INVERSE: PRINT"APPUYEZ SUR UNE TOUCHE SVP":
NORMAL: POKE-16368,0: WAIT-16384,128: POKE-16368,0:HOME
266 NEXT I
270 VTAB12: HTAB17: INVERSE : PRINT"2.PHASE":NORMAL
273 FOR I=1 TO N-1:FOR J=I+1 TO N:IF A$(I) A$(J) THEN H$=N$(I):
N$(I)=N$(J): N$(J)=H$: H$=A$(I): A$(I)=A$(J): A$(J)=H$
276 NEXT J,I
280 HOME: Z=0: FOR I=1 TO N: PRINT N$(I),A$(I): Z=Z+1
283 IF Z=M OR I=N THEN Z=0: PRINT :FOR Q=1 TO 40: PRINT"*";: NEXT:
PRINT:VTAB22:CALL-868: HTAB6: INVERSE: PRINT"APPUYEZ SUR UNE TOUCHE SVP":
```

```
NORMAL: POKE-16368,0: WAIT-16384,128: POKE-16368,0:HOME
286 NEXT I
290 VTAB12:HTAB15:INVERSE:PRINT"FIN":NORMAL:POKE34,0:PRINT:END
1000 DATA PIERRE,40,PAUL,27,JACQUES,36,LUCIEN,56
1010 DATA RENE,24,MICHEL,45,CATHERINE,34,SIMONE,28
1020 DATA FRANCOIS,55,LEON,48,LOUIS,76,CLAUDE,40
1030 DATA LUC,25,ANNE,35,MURIELLE,34,SERGE,55
1040 DATA MARTINE,44,JEROME,52,SYLVIE,22,LEOPOLD,32
```

On a choisi des numéros de ligne qui facilitent la comparaison entre ces trois versions. En principe il y a sûrement plus de programmes qui ressemblent à la version 2. Dans notre petit exemple le programme 2 est choisi pour son occupation mémoire (sans les REM dans le programme principal, la différence est encore plus apparente). Naturellement dans un exemple aussi simple les différences ne sont pas énormes , mais elles existent comme le montre le tableau suivant:

Adresses pour:	Emplacements mémoire	Versions 1	2	3
Début du programme	103/104	2049	2049	2049
Fin du programme	105/106	3123	3105	3173
Emplacements de mémoire en octets	-	1075	1057	1125
Fin de la mémoire utilisée	109/110	3459	3294	3362
Emplacements de mémoire pour des variables	-	336	189	189
Place disponible	-	1411	1246	1314

On voit que dans la version 1 avec ses nombreux sous-programmes on utilise plus de place dans les deux autres; la place est surtout occupée par les variables, le programme lui-même est également plus long que celui de la version 2, parce que l'économie réalisée avec les sous-programmes est perdue par le transfert des variables. Le gain de place est plus important car les sous-programmes plus longs sont souvent sautés. Avec APPLE si l'on doit mettre en oeuvre de très gros programmes, on n'utilise pas une technique de sous-programme aussi compliquée.
Ces exemples servent surtout a expliquer que le déroulement du programme dépend essentiellement du style de programmation.

5.3.Techniques de menu

En général, les programmes ne remplissent pas une fonction mais plusieurs et certaines les unes après les autres, d'autres étant choisies. Dans le deuxième cas, il faut que l'utilisateur puisse voir les possibilités du programme, c'est-à-dire qu'il ait une vue d'ensemble des solutions existantes. Ainsi, un programme de traitement de textes peut mémoriser un nouveau texte, transformer l'ancien ou le développer. Cette possibilité de choix affichée à l'écran est appelée: MENU. Elle correspond à la "carte" (le menu) du programme. Le CATALOG d'une disquette n'est au fond rien d'autre que cela mais, on attend d'un vrai menu un aménagement intéressant et une méthode de choix simplifiée.

Après chaque sélection l'utilisateur saute à une autre partie du programme. Pour cela on utilise les instructions de branchements conditionnels. En dehors de l'instruction IF....THEN que nous connaissons déjà, il existe en BASIC APPLESOFT deux autres instructions de branchements conditionnels, dont une possède avec RETURN un retour automatique. Ces instructions sont utilisées en général de la manière suivante:

```
-----------------------------------
I ON variable GOTO nligne 1,2,3,..  I
I---------------------------------I
I ON variable GOSUB nligne 1,2,3,...I
-----------------------------------
```

GOSUB met en place la commande RETURN (exactement comme avec le GOSUB inconditionnel), ce qui ne se produit pas avec ON...GOTO. La commande

```
ON K GOTO 100,110,120,150
```

a des effets différents selon les valeurs de K.

La valeur de K détermine le branchement.

Si K a pour valeur 1, il se produit un branchement sur la ligne 100, donc sur le premier numéro de ligne situé après GOTO; si K =2 le branchement se produit à la ligne 110, si K =3 dans la ligne 120, si K =4 dans la ligne 150. Il se produit un branchement pour chaque valeur de K plus grande ou égale à 1 et plus petite ou égale à 5. Sinon, le programme sautera à la prochaine instruction, même si elle est située sur la même ligne. C'est en cela que réside la principale différence avec la commande IF....THEN.

Pour le programmeur il existe la possibilité d'écrire plusieurs instructions dans une même ligne, c'est un petit truc qui permet d'économiser de la place parce que chaque numéro de ligne occupe de la mémoire: si après la commande IF A =1 THEN 50, aucune autre instruction ne peut être écrite à la suite, on peut utiliser l'instruction ON A GOTO 50. Pour A = 1 le branchement désiré se produit,si A 1 les instructions suivantes seront exécutées.

Pour ON....GOSUB il se passe la même chose: après le traitement du sous-programme il y a un retour sur l'instruction suivante même si elle est placée sur la même ligne.

Voici un exemple d'utilisation qui servira de modèle de menu. Il s'agit d'une impression des nombres de 1 à 16. Le programme forme une matrice 4 * 4 comprenant les nombres de 1 à 4, de 5 à 8, 9 à 12 , 13 à 16.

On dispose de quatre possibilités différentes: la manière la plus courante c'est-à-dire: 1 2 3 4 sur la première ligne, 5 6 7 8 sur la seconde, etc.....; en miroir c'est-â-dire: 4 3 2 1 sur la première ligne, 8 7 6 5 sur la seconde, etc.....; normalement mais avec la mise en évidence de la diagonale principale et enfin la dernière avec la mise en évidence de la diagonale secondaire. Le menu a donc 4 possibilités:

```
************************************
*                                  *
*  ETABLISSEMENT D'UNE MATRICE 4*4  *
*                                  *
************************************


  MODE NORMAL.....1

  MODE MIROIR.....2

  DIAGONALE PRINCIPALE..3

  DIAGONALE SECONDAIRE..4

  ENTREE (5-FIN)       ..

IL SUFFIT DE CHOISIR UN CHIFFRE !
```

Après l'entrée d'un chiffre entre 1 et 5 on a sur l'écran l'image correspondante au programme suivant:

```
3 HOME : HTAB(9):PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 DIM Z(4,4),Z$(4,4)
20 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
30 FOR I=1 TO 4: FOR J=1 TO 4: READ Z(I,J): NEXT : NEXT
40 FOR I=1 TO 4: FOR J=1 TO 4: Z$(I,J)=STR$(Z(I,J)): IF Z(I,J) 10 THEN
Z$(I,J)=" "+Z$(I,J)
50 NEXT J,I
60 HOME: HTAB4: FOR Q=1 TO 34: PRINT"*";: NEXT: PRINT: HTAB4:
PRINT"*";:HTAB37:PRINT"*"
70 HTAB4: PRINT"*";:HTAB7:PRINT"ETABLISSEMENT D'UNE MATRICE
4*4";:HTAB37:PRINT"*"
80 HTAB4: PRINT"*";: HTAB 37: PRINT"*":HTAB4: FOR I=1 TO 34:
PRINT"*";:NEXT:POKE34,6
100 HOME: VTAB10: HTAB11: PRINT"MODE NORMAL.....1":PRINT:
HTAB11:PRINT"MODE MIROIR.....2":PRINT: HTAB11:PRINT"DIAGONALE
PRINCIPALE..3":PRINT: HTAB11: PRINT"DIAGONALE SECONDAIRE..4": PRINT
110 VTAB22:HTAB3: INVERSE: PRINT"IL SUFFIT DE CHOISIR UN CHIFFRE !";:
NORMAL
120 VTAB18:HTAB11: INPUT"ENTREE (5-FIN) ..";X$: X=VAL(X$)
130 FOR I=1 TO 5:IF X I THEN NEXT: VTAB18: HTAB30: INVERSE:
PRINT"ERREUR!";: NORMAL: FOR Q=1 TO 1500: NEXT: HTAB 30: CALL -868: GOTO120
140 ON X GOTO 500,520,540,570,230
200 VTAB 23: HTAB 6: INVERSE: PRINT"APPUYEZ SUR UNE TOUCHE": NORMAL : POKE
-16368,0
210 GET A$: IF A$="" THEN210
220 POKE -16368,0: GOTO 100
230 POKE 34,0: CALL -958: END
500 REM IMPRESSION DE LA MATRICE
510 HOME : VTAB 10: FOR I=1 TO 4: FOR J=1 TO 4: HTAB(6*J+5): PRINTZ$(I,J);:
NEXT:GOTO 200
520 REM IMPRESSION DE LA MATRICE MIROIR
530 HOME : VTAB 10: FOR I=1 TO 4: FOR J=4 TO 1 STEP-1: HTAB(35-6*J): PRINT
```

```
Z$(I,J);:NEXT: PRINT: NEXT: GOTO200
540 REM IMPRESSION DE LA DIAGONALE PRINCIPALE
550 HOME : VTAB 10: FOR I=1 TO 4: FOR J=1 TO 4: HTAB(6*J+5): IF I=J THEN
INVERSE
560 PRINT Z$(I,J);: NORMAL: NEXT :PRINT : PRINT: NEXT: GOTO 200
570 REM IMPRESSION DE LA DIAGONALE SECONDAIRE
580 HOME : VTAB 10: FOR I=1 TO 4: FOR J=1 TO 4: HTAB(6*J+5): IF (I+J)=5
THEN INVERSE
590 PRINT Z$(I,J);: NORMAL: NEXT: PRINT: PRINT: NEXT: GOTO200
```

Explication rapide du programme:

Lignes	Description
10	2 champs sont doublement indicés pour dimensionner la matrice 4*4.
20-30	Lignes de DATA, et lecture des données.
40-50	Les nombres sont placés dans une chaîne. Afin de pouvoir ajouter un espace aux chiffres.
60-80	Impression du titre.
100-120	Menu. On choisit un chiffre de 1 à 5.
130	Si une valeur différente est entrée, on envoie un message d'erreur et on retourne à la ligne 120.
140	Branchement avec ON...GOTO.
200-220	Boucle d'attente conditionnelle, permet à l'utilisateur de lire l'écran.
230	Lorsque le 5 est entré pour "fin du programme" la fenêtre de l'écran se lève, le message: "CHOISISSEZ..." s'efface et le programme est terminé.
500-510	Normalement les nombres de 1 à 16 sont transférés dans une matrice 4*4 avec l'aide d'une valeur de HTAB calculée. A chaque impression on prend la chaîne Z$. Ensuite on se branche sur la boucle d'attente conditionnelle ligne 200.
520-530	Chaque impression produit une ligne, dans laquelle l'ordre est inversé; c'est pourquoi la boucle J qui fournit l'index pour la colonne a un pas de progression négatif. Puis on se branche ligne 200.
540-560	On utilise la matrice normale,les éléments de la diagonale

	principale sont en inverse.
570-590	Les éléments de la diagonale secondaire sont en inverse.

Dans cet exemple, on choisit l'option avec les chiffres de 1 à 5. L'entrée est donnée par X$ pour éviter le message REENTER qui pertuberait la présentation. Avec la fonction VAL cette chaîne est transformée en valeur numérique. Cette valeur X, permet, dans la ligne 130, d'indiquer une erreur.

Le branchement se produit grâce à l'instruction ON...GOTO, on aurait pu choisir l'instruction ON...GOSUB, les 4 parties de programme sont terminées avec les instructions de branchement GOTO 200 dans les lignes 500 à 590. Après ces instructions la ligne après l'instruction de branchement est exécutée ce qui se serait produit avec RETURN.

Avec l'exemple suivant, on entre une lettre. Les lettres de A à E sont admises dans la ligne 40 et placées dans le tableau B$ grâce à la ligne DATA 600. L'indice I délivre la variable numérique nécessaire au branchement ON...GOTO. A l'écran le menu est modifié (lignes 100 à 120).

```
3 HOME : HTAB(9):PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 DIM Z(4,4),Z$(4,4),B$(5)
20 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
30 FOR I=1 TO 4: FOR J=1 TO 4: READ Z(I,J): NEXT : NEXT
40 FOR I=1 TO 5:READ B$(I): NEXT
50 FOR I=1 TO 4: FOR J=1 TO 4: Z$(I,J)=STR$(Z(I,J)): IFZ(I,J) 10 THEN
Z$(I,J)=" "+Z$(I,J)
60 NEXT J,I
70 HOME: HTAB 4: FOR Q=1 TO 34: PRINT"*";:NEXT: PRINT: HTAB4:
PRINT"*";:HTAB37: PRINT"*"
80 HTAB4: PRINT"*";:HTAB7:PRINT"ETABLISSEMENT D'UNE MATRICE
4*4";:HTAB37:PRINT"*"
90 HTAB4: PRINT"*";: HTAB 37: PRINT"*":HTAB4: FOR I=1 TO 34:
PRINT"*";:NEXT:POKE34,6
100 HOME: VTAB10: HTAB11: PRINT"MODE NORMAL.....A":PRINT:
HTAB11:PRINT"MODE MIROIR.....B":PRINT: HTAB11:PRINT"DIAGONALE
PRINCIPALE..C":PRINT: HTAB11: PRINT"DIAGONALE SECONDAIRE..D": PRINT
110 HTAB11: PRINT"E-FIN"
120 VTAB 20:HTAB16: INVERSE :PRINT"CHOISISSEZ": PRINT: HTAB6: NORMAL:
PRINT" UNE LETTRE !";
130POKE -16368,0: GET X$: POKE 16368,0
140 FOR I=1 TO 5:IF X$ B$(I) THEN NEXT I: VTAB18: HTAB32: INVERSE:
PRINT"ERREUR!";: NORMAL: FOR Q=1 TO 1500: NEXT Q: HTAB 32: CALL -868:
GOTO120
150 POKE 37,(6+2*I): PRINT :POKE 36,29: FLASH: PRINT B$(I): FOR Q=1 TO
1500: NEXT: NORMAL
160 ON I GOTO 500,520,540,570,230
200 VTAB 23: HTAB 6: INVERSE: PRINT"APPUYEZ SUR UNE TOUCHE": NORMAL : POKE
-16368,0
210 GET A$: IF A$="" THEN210
220 POKE -16368,0: GOTO 100
230 POKE 34,0: POKE 37,16:POKE 36,29: PRINT"E": CALL -958: END
```

```
500 REM IMPRESSION DE LA MATRICE
510 HOME : VTAB 10: FOR I=1 TO 4: FOR J=1 TO 4: HTAB(6*J+5): PRINTZ$(I,J);:
NEXT:GOTO 200
520 REM IMPRESSION DE LA MATRICE MIROIR
530 HOME : VTAB 10: FOR I=1 TO 4: FOR J=4 TO 1 STEP-1: HTAB(35-6*J):  PRINT
Z$(I,J);:NEXT: PRINT: NEXT: GOTO200
540 REM IMPRESSION DE LA DIAGONALE PRINCIPALE
550 HOME : VTAB 10: FOR I=1 TO 4: FOR J=1 TO 4: HTAB(6*J+5):  IF  I=J  THEN
INVERSE
560 PRINT Z$(I,J);: NORMAL: NEXT :PRINT : PRINT: NEXT: GOTO 200
570 REM IMPRESSION DE LA DIAGONALE SECONDAIRE
580 HOME : VTAB 10: FOR I=1 TO 4: FOR J=1 TO  4:  HTAB(6*J+5):  IF  (I+J)=5
THEN INVERSE
590 PRINT Z$(I,J);: NORMAL: NEXT: PRINT: PRINT: NEXT: GOTO200
600 DATA A,B,C,D,E
```

Maintenant l'entrée se fait par GET X$ (ligne 130), les entrées non permises sont rejetées par la comparaison de X$ et B$ (I) et un message est envoyé. La ligne 140 est un peu différente de la ligne 130 du programme précédent; la ligne 150 est entièrement nouvelle. Lorsque dans la boucle de la ligne 140, l'index est déterminé, le curseur est positionné sur la lettre correspondante du menu, avec les 2 commandes POKE aux emplacements de mémoire 37 et 36. La lettre clignote. Après la boucle d'attente, le branchement a lieu sur la ligne 160. Le programme restant est identique à la première version.

La troisième version du programme ne se différencie des deux précédentes que par la manière dont la sélection est opérée, donc seulement dans les lignes 100 à 150; et en particulier dans les lignes 130 à 140. Celles-ci déterminent la valeur de I pour le branchement. Avec la boucle P dans la ligne 130 le curseur se déplace vers la gauche de lignes en lignes, en commençant toujours par la première possibilité de sélection. Ceci se produit jusqu'à ce que l'on appuie sur une touche quelconque. Le contenu de l'emplacement mémoire -16384 doit être pour cela transféré dans la boucle. Le curseur se trouve dans la position P et la valeur I correspondante est calculée en ligne 140.

```
3 HOME : HTAB(9):PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 DIM Z(4,4),Z$(4,4)
20 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
30 FOR I=1 TO 4: FOR J=1 TO 4: READ Z(I,J): NEXT : NEXT
40 FOR I=1 TO 4: FOR J=1 TO 4: Z$(I,J)=STR$(Z(I,J)): IFZ(I,J) 10 THEN
Z$(I,J)=" "+Z$(I,J)
50 NEXT J,I
60 HOME: HTAB 4: FOR Q=1 TO 34: PRINT"*";:NEXT: PRINT: HTAB4:
PRINT"*";:HTAB37: PRINT"*"
70 HTAB4: PRINT"*";:HTAB7:PRINT"ETABLISSEMENT D'UNE MATRICE
4*4";:HTAB37:PRINT"*"
80 HTAB4: PRINT"*";: HTAB 37: PRINT"*":HTAB4: FOR I=1 TO 34:
PRINT"*";:NEXT:POKE34,6
100 HOME: VTAB10: HTAB11: PRINT"1.MODE NORMAL":PRINT: HTAB11:PRINT"2.MODE
MIROIR":PRINT: HTAB11:PRINT"3.DIAGONALE PRINCIPALE":PRINT: HTAB11:
PRINT"4.DIAGONALE SECONDAIRE": PRINT
110 HTAB11: PRINT"5.FIN"
120 VTAB 20:HTAB16: INVERSE :PRINT"CHOISISSEZ": PRINT: HTAB6: NORMAL:
PRINT" UNE LETTRE !";:POKE -16368,0
130 FOR P=10 TO 18 STEP2: VTAB P: HTAB2: INVERSE: PRINT" ";:FOR Q=1 TO 750:
NEXTQ: IF PEEK(-16384) 128 THEN HTAB2: NORMAL: PRINT" ": NEXT P:GOTO 130
140 I=P/2-4 : NORMAL: HTAB2: PRINT " "
150 ON I GOTO 500,520,540,570,230
200 VTAB 23: HTAB 6: INVERSE: PRINT"APPUYEZ SUR UNE TOUCHE": NORMAL : POKE
-16368,0
210 GET A$: IF A$="" THEN210
220 POKE -16368,0: GOTO 100
230 POKE -16368,0: POKE 34,0: CALL -958: END
500 REM IMPRESSION DE LA MATRICE
510 HOME : VTAB 10: FOR I=1 TO 4: FOR J=1 TO 4: HTAB(6*J+5): PRINTZ$(I,J);:
NEXT:GOTO 200
520 REM IMPRESSION DE LA MATRICE MIROIR
```

```
530 HOME : VTAB 10: FOR I=1 TO 4: FOR J=4 TO 1 STEP-1: HTAB(35-6*J): PRINT
Z$(I,J);:NEXT: PRINT: NEXT: GOTO200
540 REM IMPRESSION DE LA DIAGONALE PRINCIPALE
550 HOME : VTAB 10: FOR I=1 TO 4: FOR J=1 TO 4: HTAB(6*J+5): IF I=J THEN
INVERSE
560 PRINT Z$(I,J);: NORMAL: NEXT :PRINT : PRINT: NEXT: GOTO 200
570 REM IMPRESSION DE LA DIAGONALE SECONDAIRE
580 HOME : VTAB 10: FOR I=1 TO 4: FOR J=1 TO 4: HTAB(6*J+5): IF (I+J)=5
THEN INVERSE
590 PRINT Z$(I,J);: NORMAL: NEXT: PRINT: PRINT: NEXT: GOTO200
```

On peut utiliser des menus pour tous les programmes devant résoudre plusieurs problèmes. Comme nous l'avons montré dans ces trois exemples, il existe plusieurs possibilités. Elles peuvent servir de point de départ pour l'élaboration d'autres formes de menu.

5.4 Fusion de programmes

Souvent on aimerait ne pas avoir à écrire à nouveau un sous-programme déjà utilisé dans un ancien programme. C'est la possibilité qu'offre les fichiers de textes EXEC. Ce sont des fichiers séquentiels qui contiennent des commandes et des lignes de programme. Leur mise en oeuvre est un peu complexe, nous donnerons donc quelques exemples simples.

Lorsqu'on désire connaître la place occupée par un programme, il faut interroger toute une série de pointeurs que l'on peut comparer selon la version du programme (cf. 5.2) mais il est un peu fastidieux de toujours devoir entrer les même suites d'instructions. On peut mémoriser ces instructions dans un fichier de texte et ainsi on peut les appeler lorsque cela est nécessaire grâce à la commande EXEC. Le programme suivant appelé MEMOIRE permet ce stockage:

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 D$=CHR$(4): REM CTRL-D POUR ADRESSAGE DISQUETTE
15 Q$=CHR$(34):REM GUILLEMETS
20 PRINT D$;"OPEN MEMOIRE"
30 PRINT D$;"WRITE MEMOIRE"
40 PRINT"?";Q$;"103: ";Q$;";PEEK(103)+PEEK(104)*256"
50 PRINT"?";Q$;"105: ";Q$;";PEEK(105)+PEEK(106)*256"
60 PRINT"?";Q$;"109: ";Q$;";PEEK(109)+PEEK(110)*256"
70 PRINT"?";Q$;"111: ";Q$;";PEEK(111)+PEEK(112)*256"
80 PRINT"?";Q$;"115: ";Q$;";PEEK(115)+PEEK(116)*256"
90 PRINT D$;"CLOSE"
100 END
```

Après RUN, un fichier séquentiel est ouvert avec le nom MEMOIRE en écriture. Les commandes PRINT des lignes 40 à 80 ne permettent aucune

impression d'écran mais le stockage sur disquette. Les commandes entre guillemets sont mémorisées. Avec l'instruction PRINT chaque ligne de commande donne une adresse mémoire et son contenu. Une ligne de commande doit être placée entre guillemets avec la fonction CHR$ pour éviter tout malentendu.

On trouve, alors, les lignes suivantes dans le fichier:

```
PRINT"103: ";PEEK(103)+PEEK(104)*256
PRINT"105: ";PEEK(105)+PEEK(106)*256
PRINT"109: ";PEEK(109)+PEEK(110)*256
PRINT"111: ";PEEK(111)+PEEK(112)*256
PRINT"115: ";PEEK(115)+PEEK(116)*256
```

Après la mémorisation du programme on connaît l'occupation mémoire grâce à EXEC MEMOIRE. Cette commande produit le même effet qu'une entrée au clavier.

Le deuxième exemple montre un fichier de texte: IMPRESSION, qui contient les commandes et les lignes de programme:

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 D$=CHR$(4) : Q$=CHR$(34)
20 PRINT D$;"OPEN IMPRESSION"
30 PRINT D$;"WRITE IMPRESSION"
40 PRINT "3 HOME:HTAB(9):PRINT ";Q$;"APPLE TRUCS & ASTUCES";Q$
50 PRINT "4 PRINT:PRINT:HTAB(9):PRINT ";Q$;"R.PRUST / W.VOSS
1984";Q$;":PRINT"
60 PRINT"PR 1"
70 PRINT"PRINT CHR$(9);";Q$;"80N";Q$
80 PRINT"PRINT CHR$(9);";Q$;"10L";Q$
90 PRINT"PRINT CHR$(9);";Q$;"65R";Q$
```

```
100 PRINT"PRINT CHR$(9);";Q$;"65P";Q$
110 PRINT"LIST-3"
120 PRINT"PR 0"
130 PRINT D$;"CLOSE IMPRESSION"
140 END
```

Avec les lignes 40 et 50, deux lignes de programme sont écrites dans le fichier avec les numéros 3 et 4. Il faut de nouveau mettre des guillemets avec CHR$(34).

Les lignes 60 à 120 écrivent des commandes dans le fichier dont le contenu se compose de deux lignes de programme et de quelques commandes pour l'imprimante:

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
PR 1
PRINT CHR$(9);"80N"
PRINT CHR$(9);"10L"
PRINT CHR$(9);"65R"
PRINT CHR$(9);"65P"
LIST-3
PR 0
```

Si ce fichier est appelé par EXEC IMPRESSION, les lignes 3 et 4 sont ajoutées au programme en mémoire qui n'est donc pas perdu comme avec la commande RUN. De plus, les commandes imprimantes sont exécutées et le nouveau programme est sorti sur imprimante.

Une possibilité plus simple d'ajouter des lignes à un programme est d'utiliser le programme MAKE LIST:

```
1 GOTO 10
2 PR 1:PRINT CHR$(9);"80N":PRINT CHR$(9);"10L":PRINT CHR$(9);"65R":
PRINTCHR$(9) ;"65P": LIST3-: PR 0: END
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 D$=CHR$(4)
20 PRINT D$;"OPEN LIST"
30 PRINT D$;"WRITE LIST"
40 LIST 2-4
50 PRINT "R''N"
60 PRINT "DEL 2,4"
70 PRINT D$;"CLOSE LIST"
80 TEXT: END
```

Le contenu de LIST est le suivant:

```
2 PR 1:PRINT CHR$(9);"80N":PRINT CHR$(9);"10L":PRINT CHR$(9);"65R":
PRINT CHR$(9) ;"65P": LIST3-: PR 0: END
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
```

```
RUN
DEL 2,4
```

Après la commande EXEC LIST les lignes 2 à 4 seront ajoutées au programme en mémoire à la condition suivante: le programme d'origine ne doit contenir que des numéros de lignes supérieurs à 4. Après RUN on met en route le programme élargi c'est-à-dire que l'on traite la ligne 2 avec les instructions de l'imprimante. Puis le contrôle revient à LIST qui détruit les lignes 2 à 4. Si on a besoin d'un listing sur imprimante, pour rechercher par exemple une erreur, on entre EXEC LIST qui donne un listing

avec 2 lignes supplémentaires et le programme sous sa forme précédente de manière à pouvoir continuer le travail.

Une autre possibilité permet de créer des fichiers permettant de fusionner plusieurs parties de programmes qui ont déjà été utilisées dans d'autres circonstances. Certains sous-programmes reviennent souvent comme certaines impressions. Il est intéressant d'écrire un programme qui crée des fichiers avec les lignes de programme voulues, le premier numéro étant choisi comme on le désire.

Le problème du tri de deux tableaux, l'un numérique et l'autre alphanumérique est souvent rencontré. C'est pourquoi nous l'avons choisi comme exemple de démonstration. Le programme suivant est en mémoire:

```
3 HOME : HTAB(9): PRINT"APPLE TRUCS & ATUCES"
4 PRINT: PRINT :PRINT TAB(9)"R.PRUST / W.VOSS 1984"
10 REM CONSTRUCTION D'UN SOUS-PROGRAMME
20 N=20: M=5: TEXT: HOME :DIM N$(N),A$(N)
30 GOTO 100
40 VTAB22: CALL -868: HATB6: INVERSE: PRINT"SVP APPUYEZ SUR UNE TOUCHE":
NORMAL:POKE -16368,0: WAIT -16384,128: POKE -16368,0: HOME: RETURN
50 FOR Q=1 TO 40: PRINT"*";: NEXT: PRINT: RETURN
100 REM DEBUT DU PROGRAMME
110 FOR I=1 TO N: READ N$(I),A$(I): NEXT
120 GOSUB 200
130 HOME : Z=0: FOR I=1 TO N: PRINT N$(I),A$(I): Z=Z+1: IF Z=M OR I=M THEN
Z=0: PRINT: GOSUB 50: GOSUB 40: REM TRAIT ET ATTENDRE
140 NEXT I
150 VTAB 12: HTAB 15:INVERSE: PRINT"FIN": NORMAL:POKE34,0: PRINT: END
```

C'est le programme principal avec deux petits sous-programmes (lignes 40 à 50) qui lit les valeurs dans deux tableaux de chaînes; les chaînes d'un tableau doivent être classés alphabétiquement. Toutes les permutations à l'intérieur du premier tableau doivent se produire dans le deuxieme tableau. C'est à cela que sert le sous-programme de la ligne 200 appelé en 120.
Ceci se présente sur disquette de la maniere suivante:

```
10 FOR I=1 TO N-1: FOR J=I+1 TO N: IF N$(I) N$(J) THEN H$=N$(I):
N$(I)=N$(J): N$(J)=H$: H$=A$(I): A$(I)=A$(J): A$(J)=H$
20 NEXT J,I
30 RETURN
```

On a maintenant la possibilité de retaper les lignes avec d'autres numéros de lignes; ou on peut numéroter grâce à la touche ESC (voir chapitre 6). Il est possible encore d'utiliser le programme MAKE TRI qui permet d'utiliser des numéros de lignes quelconque.

Le programme principal est mémorisé sur disquette, puis, on tape MAKE TRI:

```
3 HOME : HTAB(9): PRINT"APPLE TRUCS & ATUCES"
4 PRINT: PRINT :PRINT TAB(9)"R.PRUST / W.VOSS 1984"
10 REM CONSTRUCTION D'UN SOUS-PROGRAMME
20 REM AVEC DES NUMEROS DE LIGNES QUELCONQUES
30 INPUT"PREMIER NUMERO DE LIGNE ";ZN
40 D$=CHR$(4)
50 PRINT D$;"OPEN TRI"
60 PRINT D$;"DELETE TRI"
70 PRINT D$;"OPEN TRI"
80 PRINT D$;"WRITE TRI"
90 PRINT ZN;"FOR I=1 TO N-1: FOR  J=I+1 TO N: IF N$(I) N$(J) THEN H$=N$(I):
N$(I)=N$(J): N$(J)=H$: H$=A$(I): A$(I)=A$(J): A$(J)=H$": ZN=ZN+1
100 PRINT ZN;"NEXT J,I":ZN=ZN+1
110 PRINT ZN;"RETURN"
120 PRINT D$;"CLOSE TRI"
130 END
```

Le premier numéro de ligne du sous-programme est demandé. Eventuellement, l'ancien fichier de texte TRI est détruit ce qui permet d'en ouvrir un autre en écriture.
Avec les instructions PRINT, les lignes 90 à 110 deviennent les lignes du programme écrites dans le fichier TRI. La variable ZN permet d'établir les numéros de lignes et on incrémente ZN pour chaque nouvelle ligne; TRI est composé ainsi:

```
200 FOR I=1 TO N-1: FOR J=I+1 TO N: IF N$(I) N$(J) THEN H$=N$(I):
N$(I)=N$(J): N$(J)=H$: H$=A$(I): A$(I)=A$(J): A$(J)=H$
201 NEXT J,I
202 RETURN
```

Avec RUN MAKE TRI (au cas où le programme se trouve déjà sur la disquette), le fichier TRI est stocké. Ensuite le programme principal est de nouveau mémorisé et on peut compléter les lignes du sous-programme avec la commande EXEC.

L'écriture du programme pour l'établissement des fichiers de textes demande un travail une attention particulière car dans les instructions PRINT ondevra placer des guillemets, sous la forme CHR$(34). Ceci a déjà été montré dans les deux premiers exemples de cette partie. On trouvera un autre exemple très important dans le chapitre 7.5 .

Ce procédé peut également servir à simplifier la création d'un menu. Souvent les lignes d'un programme sont construites de la même manière pour pouvoir utiliser un générateur de lignes de programme . L'inconvénient le plus important réside dans le fait qu'il faut mémoriser plusieurs fois et il n'est pas possible de travailler dans le programme principal. En effet si on n'a pas établi auparavant MAKE TRI le programme est perdu.

L' exemple important qui porte sur l'établissement d'un listing sur imprimante et en même temps sur les possibilités de formation d'un fichier

EXEC.

Le programme en mémoire peut être listé partiellement ou entièrement, sans que l'utilisateur envoie les commandes par le clavier. EXEC LISTE est la dernière commande à taper. Elle appelle un fichier de texte qui mettra en oeuvre le programme MAKE LISTE:

```
3 HOME : HTAB(9): PRINT"APPLE TRUCS & ATUCES"
4 PRINT: PRINT :PRINT TAB(9)"R.PRUST / W.VOSS 1984"
10 D$=CHR$(4): Q$=CHR$(34)
20 PRINT D$;"OPEN LISTE"
30 PRINT D$;"DELETE LISTE"
40 PRINT D$;"OPEN LISTE"
50 PRINT D$;"WRITE LISTE"
60 PRINT"PEEK(175)+PEEK(176)*256"
70 PRINT"Q=PEEK(103)+PEEK(104)*256: X1=PEEK(Q+2)+PEEK(Q+3)*256"
80 PRINT"FOR Q=Q TO Y: Z =PEEK(Q) +PEEK(Q+1)*256: IF PEEK(Z)
+PEEK(Z+1)*256 0 THEN Q=Z-1: NEXT Q"
90 PRINT"Z2=PEEK(Q+2)+PEEK(Q+3)*256"
100 PRINT"Z1=X1: Z2=X2"
110 PRINT"X=PEEK(115)+PEEK(116)*256"
120 PRINT"T=INT(Z1/256): POKEX-4,T: POKE X-5,Z1-T*256"
130 PRINT"T=INT(Z2/256): POKEX-2,T: POKE X-3,Z2-T*256"
140 PRINT"T=INT((X-6)/256): POKE116,T: POKE 115,X-6-T*256"
150 PRINT"EXEC LST"
160 PRINT CHR$(4);"CLOSE LISTE"
170 END
```

Le fichier de texte contient la suite de commandes suivantes:

```
Y=PEEK(175)+PEEK(176)*256
Q=PEEK(103)+PEEK(104)*256: X1=PEEK(Q+2)+PEEK(Q+3)*256
FOR Q=Q TO Y:Z=PEEK(Q)+PEEK(Q+1)*256: IF PEEK(Z)+PEEK(Z+1)*256 0 THEN
Q=Z-1: NEXT Q
X2=PEEK(Q+2)+PEEK(Q+3)*256
Z1=X1: Z2=X2
X=PEEK(115)+PEEK(116)*256
T=INT(Z1/256): POKE X-4,T: POKE X-5,Z1-T*256
T=INT(Z2/256): POKE X-2,T: POKE X-3,Z2-T*256
T=INT((X-6)/256): POKE 116,T: POKE 115,X-6-T*256
EXEC LST
```

Avec les commandes PEEK et POKE, on recherche le plus petit et le plus grand numéro de ligne (Z2 et Z1). Le numéro de ligne le plus petit Z1 se trouve après le début du programme aux emplacements de mémoire 2 et 3, il est recherché avec les adresses 103 et 104. On mémorise chaque ligne de la manière suivante: les deux premiers octets contiennent l'adresse de la prochaine ligne (partie basse d'abord); les deux octets suivants contiennent le numéro de ligne, ensuite, le texte est mémorisé. On procède de la même manière pour déterminer le numéro de ligne Z2. A l'aide du pointeur, l'ordinateur passe rapidement d'une ligne à l'autre , jusqu'à ce que il trouve un emplacement de mémoire contenant zéro. A l'emplacement Q ainsi trouvé, les dernières lignes sont mémorisées et aini le numéro de ligne le plus grand se trouve dans les emplacements Q + 2 et Q + 3 .

Pour que les valeurs Z1 et Z2 ne soient pas perdues pendant le déroulement, elles sont mémorisées directement sous HIMEM (emplacements de mémoire 115-116). Ensuite on abaisse HIMEM pour protéger ces valeurs.

La dernière instruction du fichier de texte LISTE appelle le fichier EXEC

LST. Ce qui a été également utilisé avec MAKE LST:

```
3 HOME : HTAB(9): PRINT"APPLE TRUCS & ATUCES"
4 PRINT: PRINT :PRINT TAB(9)"R.PRUST / W.VOSS 1984"
10 REM ETABLISSEMENT D'UN EXEC'S
20 REM POUR LE LISTING IMPRIMANTE
30 D$=CHR$(4)
40 PRINT D$;"OPEN LST"
50 PRINT D$;"WRITE LST"
60 PRINT "SAVE INTER"
70 PRINT "RUN ENTREE"
80 PRINT "LOAD INTER"
90 PRINT "DELETE INTER"
100 PRINT "EXEC LSTR"
110 PRINT D$;"CLOSE LST"
```

Le fichier ainsi établi prend le contrôle du déroulement. Dans un fichier de texte la commande EXEC doit être la dernière car le premier fichier de texte est fermé avant le début de la réalisation du second. LST contient les commandes suivantes:

```
SAVE INTER
REM ENTREE
LOAD INTER
DELETE INTER
EXEC LSTR
```

Tout d'abord le programme qui doit être listé est stocké sur une disquette avec le nom: INTER, ensuite, le programme ENTREE est mis en route, il met également en oeuvre un fichier texte LSTR.

Le programme de départ est de nouveau mémorisé, la copie temporaire est détruite, le fichier de texte comprenant les commandes suivantes est

appelé:

```
122 PR 1: PRINT CHR$(9);"80N"
123 PRINT CHR$(9);"10L"
124 PRINT CHR$(9);"65R"
125 PRINT CHR$(9);"65P"
126 LIST 20-40: PR 0: END
RUN 122
DEL 122,126
```

Les numéros de lignes apparaissant dans le fichier dépendent du programme à lister et des entrées de l'utilisateur. Avec la commande EXEC LSTR cinq nouvelles lignes sont ajoutées derrière le programme initial, ceci est produit par le listing de l'imprimante, puis les lignes ajoutées sont effacées pour que le programme reste dans la mémoire centrale sous sa forme initiale. Les numéros de lignes 20 à 40 qui doivent être listés ne sont qu'un exemple. Les numéros de ligne seront déterminés avec l'aide du programme ENTREE, qui établit également le fichier texte LSTR:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: PRINT TAB(9)"R.PRUST / W.VOSS 1984"
10 REM ETABLISSEMENT D'UNE IMPRESSION EXEC'S
20 D$=CHR$(4):Q$=CHR$(34)
30 T=PEEK(115)+PEEK(116)*256
40 Z1=PEEK(T+1)+PEEK(T+2)*256
50 Z2=PEEK(T+3)+PEEK(T+4)*256: X2=Z2
60 TT=INT((T+6)/256): POKE 115,T+6-TT*256: POKE 116,TT
70 HOME: VTAB 5: PRINT"DEBUT D'IMPRESSION";
80 FOR I=1 TO 6: FLASH: PRINT CHR$(160);:POKE -16368,0: WAIT -16384,128
90 NORMAL :POKE 36,PEEK(36-1)
100 X=PEEK(-16384): IF X  141 THEN X$=X$+CHR$(X-1228): PRINT CHR$(X);: NEXT
I
110 PRINT CHR$(160): POKE -16368,0: IF X$  "" THEN Z1=VAL(X$)
120 VTAB 8: PRINT"FIN D'IMPRESSION";
130 X$="": FOR I=1 TO 6:  FLASH:  PRINT  CHR$(160);:  POKE  -16368,0:  WAIT
-16384,128
140 NORMAL :POKE 36,PEEK(36)-1
150 X=PEEK(-16384): IF X  141 THEN X$=X$+CHR$(X-128):PRINT CHR$(X);: NEXT
160 PRINT CHR$(160): POKE -16368,0: IF X$  "" THEN Z2=VAL(X$)
170 PRINT D$;"OPEN LSTR"
180 PRINT D$;"DELETE LSTR"
190 PRINT D$;"OPEN LSTR"
200 PRINT D$;"WRITE LSTR"
210 PRINT X2+2;"PR 1: PRINT CHR$(9);";Q$;"80N";Q$
220 PRINT X2+3;"PRINT CHR$(9);";Q$;"10L";Q$
230 PRINT X2+4;"PRINT CHR$(9);";Q$;"65R";Q$
240 PRINT X2+5;"PRINT CHR$(9);";Q$;"65P";Q$
250 PRINT X2+6;"LIST";Z1;"-";Z2;":PR 0:END"
260 PRINT "RUN";X2+2
270 PRINT "DEL";X2+2;",";X2+6
280 PRINT D$;"CLOSE LSTR"
290 END
```

Voyons la descrition du programme:

Lignes	Contenu
20	Détermination de la chaîne Q$ pour le fichier- utilisée en 170 et suivantes.
30	La variable T est établie en fonction de HIMEM qui a été réduite avec LISTE.
40-50	Dans les quatre emplacements suivants se trouvent les valeurs de Z1 et Z2 correspondants au plus petit et au plus grand numéro de ligne
60	HIMEM est remis à sa valeur d'origine (T+6) La variable X2 donnera plus tard les valeurs pour les numéros de lignes du fichier LSTR.
70-160	L'utilisateur peut préciser la partie de programme qui doit être mise en listing. Ceci n'est pas réalisable avec un INPUT car il lirait le prochain champ du fichier 70- Message 80- Boucle pour l'entrée des numéros de lignes Six entrées sont possibles. Pour simuler l'INPUT un curseur est placé derrière le message. Le programme s'interrompt jusqu'à ce qu'une entrée au clavier ait lieu. 90- Le chiffre entré est commuté en mode NORMAL le curseur est décalé sur la gauche 100- On vérifie si RETURN a été entré, sinon les caractères entrés sont stockés dans X$ 110- Dès que le code 141 est entré, on imprime un retour chariot. X$ est complet. L'espace clignotant est remis en mode NORMAL et le tampon clavier est vidé. Si X$ n'est pas une chaîne vide, Z1 est égal à X$; autrement il reste à la valeur du plus petit numéro de

ligne.

120-160 Suite de commandes pour l'entrée des dernières lignes à lister.

170-280 Etablissement d'un fichier de texte LSTR. Les suites de commandes ont dëjâ été vues dans les exemples précédents A chaque déroulement du programme ENTREE les variables X2, Z1 et Z2 doivent pouvoir avoir des valeurs différentes.

Nous terminons cette partie avec cet exemple. Souvent les fichiers EXEC demandent un temps d'introduction très long et il est préférable pour le lecteur de ne pas prendre cet exemple comme modèle. Toutes les possibilités, des instructions INPUT ou GET n'auront pas été vues, mais, néanmoins, on pourra utiliser toutes les commandes des fichiers de texte.

Chapitre 6: Edition

6.1 Remarques préliminaires.

Dans la programmation une part importante du temps est utilisée à la correction des erreurs. Cela va de la faute de frappe à l'erreur de programmation, en passant par l'ajout de commandes supplémentaires nécessaires au développement d'un programme, ou seulement à son amélioration par le rejet des entrées erronées ou l'amélioration de la présentation.

Il est difficile de retaper les lignes car cela peut occasionner de nouvelles fautes de frappe. Si plusieurs instructions sont groupées sur une seule ligne, il est inutile de tout retaper. On pourra effectuer ces corrections facilement, même si l'APPLE ne possède pas d'éditeur implémenté avec ses propres commandes. On dispose de deux possibilités: la touche ESC, ou le programme (plus confortable) LINE EDITOR.

6.2.Le mode ESCAPE

Dans ce mode, certaines touches ont des fonctions précises. Comme nous le savons déjà, ce sont les touches de A à F et de I à M.
Le curseur se déplace de la manière suivante:

Curseur 1 ligne vers le haut - I, D
Curseur 1 ligne vers la gauche - J, B
Curseur 1 ligne vers la droite - K, A
Curseur 1 ligne vers le bas - M, C

Après ESC D on quitte le mode ESCAPE, mais pas après ESC I.
On utilisera le mode ESCAPE plutôt avec les lettres de la première colonne. On reste alors en mode ESCAPE jusqu'à ce que ESC I, J, K, ou M soit entré. On peut connaître le déplacement du curseur selon la position des touches sur le clavier.

Avec ESC E on efface la partie droite d'une ligne après le curseur; avec ESC F on efface la partie d'écran après le curseur. Dans les deux cas on reste en mode ESCAPE. Dans ce mode seuls les mouvements sur l'écran sont exécutés, et des modifications de l'écran ou du contenu de la mémoire peuvent se produire grâce aux touches "Flèche":

-- (Backspace) et -- (Retype)

Cette ligne de programme va servir d'illustration aux fonctions d'édition:

```
100 PRINT "LE RESELTAT EST EGAL A";X: IF N=5 THEN END
```

On désire faire les modifications suivantes:

1- Correction de la faute de frappe dans RESULTAT.

2- Après EGAL A on désire ajouter 2 points et un espace.

On amène d'abord la ligne sur l'écran avec LIST 100. Puis on passe en mode ESCAPE avec ESC I, pour monter d'une ligne, on appuie sur I et sur J pour amener le curseur au début de la ligne sur le 1 de 100.
Avec -- on se déplace du début de la ligne 100 jusqu'au signe erroné dans le mot RESULTAT, l'erreur est corrigée, on tape le bon signe et on utilise la touche Retype pour que le restant du mot et le E de EST soient validés. Le reste de la première ligne est composée d'espaces ce qui correspond à la représentation de l'écran de l'APPLE. Les espaces dans une chaîne qui sont placés à la fin d'une ligne doivent être sautés en mode ESCAPE.
Avec ESC K on arrive après plusieurs utilisations au S de EST qui doit être remis dans la ligne corrigée. On appuie 9 fois sur la touche Retype, et le curseur se retrouve derrière A. On peut introduire deux signes en procédant de la manière suivante:

ESC M	On descend d'une ligne.
J J	2 signes vers la gauche.
ESPACE	On suspend le mode ESCAPE
: ESPACE	Entrée des 2 signes supplémentaires
ESC I	Retour à l'ancien emplacement dans la ligne de programme c'est-à-dire entre les guillemets.

Avec la répétition de la touche Retype on peut lire maintenant le reste de la ligne, la nouvelle ligne de programme est terminée par RETURN.

```
100 PRINT "LE RESULTAT EST EGAL A: ";X:IF N=5 THEN END
```

L'édition peut être facilitée de cette manière. Lorsque des chaînes sont contenues dans une ligne on se sert de la commande

```
POKE 33,33
```

que l'on connaît depuis le paragraphe 3.1. La largeur de la fenêtre de l'écran est réduite de 40 à 33 colonnes et les espaces superflus n'apparaissent plus sur l'écran après LIST 100. Avant POKE il faut envoyer une commande HOME, pour nettoyer le reste de l'écran.

Le mode ESCAPE s'utilise également pour résumer des lignes. La ligne 100 pourrait être séparée en 2 lignes:

```
100 PRINT: PRINT"LE RESULTAT EST EGAL A: ";X
110 IF N = 5 then END.
```

Après HOME et POKE 33,33, on entre LIST 100-110. Ensuite on amène le curseur avec ESC I sur le 1 de la ligne 100. Avec la touche Retype on lit la ligne jusqu'au X. Si maintenant on tape ":" après ESC M et J, le curseur se déplace sur le I du IF, on peut relire le contenu de la ligne avec Retype. Après LIST 100-110 on a donc:

```
100 PRINT: PRINT"LE RESULTAT EST EGAL A: ";X: IF N=5 THEN END
110 IF N = 5 then END.
```

Maintenant la ligne 110 peut être détruite. On peut faire maintenant le contraire:

```
    LIST 100
on tape le numéro de ligne 110
MODE ESCAPE: le curseur est positionné sur le I de IF ligne 100
TOUCHE RETYPE (jusqu'au END)
    RETURN
```

Maintenant il y a de nouveau une ligne 110 qui contient les instructions IF...THEN. On peut raccourcir la ligne 100:

```
    LIST 100
```

MODE ESCAPE: le curseur est placé sur le 1 de 100
TOUCHE RETYPE (jusqu'au X)
RETURN

Ainsi à partir des 2 points on élimine tous les signes parce qu'ils ne sont pas relus par RETYPE, mais ils sont "sautés" par ESC K. Une fois l'édition terminée, l'écran est rétabli à sa largeur normale avec la commande

POKE 33,40

Avec le mode ESCAPE on a donc toutes les possibilités de transformer les lignes d'un programme, de les résumer ou de les sauter. Ce n'est pas aussi pratique qu'un éditeur mais on peut s'en servir pour économiser du temps après s'être exercé.

6.3. Le programme LINE EDITOR

Le programme appelé PLE ou EDITOR avec copyright de Neil Konzen permet les transformations, raccourcissements ou allongements de lignes de programme. Le programme ESC CREATE offre la possibilité d'appeler une commande fréquemment utilisée avec ESCAPE et de mettre certaines touches en mode ESCAPE. Ainsi avec ESC 1 le CATALOG de la disquette 1 peut être appelé, avec ESC 2 celui du DRIVE 2. C'est-à-dire qu'au lieu de se servir de dix touches, on ne s'en sert que de deux. ESC 1 a la même signification que CATALOG D1 RETURN. Ceci peut être transformé selon le désir de l'utilisateur.
Ces fonctions ne sont effectives que si PLE est chargé, ce qui est fait avec

```
RUN PROGRAM LINE EDITOR.
```

L'éditeur est à l'abri de l'écrasement grâce à la réduction de HIMEM, il est disponible après la commande FP et perdu après le BOOT PR 6. On donnera d'autres explications dans le paragraphe 7.3.

EDITOR est appelé par CTRL E. Le mot EDIT apparaît sur l'écran, derrière lui il faut entrer le numéro de la ligne désiré; après RETURN la ligne apparaît, le curseur est placé sur le premier caractère derrière le numéro de ligne. Les caractères de commandes sont en inverse. Dans ce mode d'édition certains caractères de commandes ont des fonctions particulières: modifier des lignes c'est-à-dire effacer des signes, en inclure ou se déplacer à l'intérieur de ces lignes.
BACKSPACE et RETYPE fonctionnent de la manière habituelle on efface les erreurs en remplaçant par le bon caractère.

Les commandes les plus importantes d'EDITOR vont être données et expliquées dans un exemple.

CTRL I : Insert - Passage en mode d'insertion.

On peut insérer un ou plusieurs signes devant le curseur. La partie à droite de la ligne est poussée sur la droite. Le mode d'insertion se termine avec RETYPE, RETURN ou une autre commande.

CTRL O : Override - Passage en mode d'insertion.

C'est la même chose que pour les insertions avec CTRL I. Après CTRL O les caractères de commandes peuvent également être insérés dans une ligne de programme. C'est après l'insertion de ce caractère que le mode d'insertion est terminé. (Par exemple CTRLD pour une commande de disquettes.)

CTRL D : Delete

Au curseur un signe est effacé et le texte à sa droite est déplacé vers la gauche.

CTRL Z : Zap.

caract. Lorsqu'on entre CTRL Z et le caractère désiré, tous les signes sont effacés à partir de la position du curseur jusqu'au premier caractère choisi qui lui sera conservé. En répétant l'entrée du signe on reproduit l'effacement.
Le signe peut être un caractère de commande.

CTRL P : Pack.

Avec cette commande tous les espaces en dehors d'une chaîne sont effacés. Peu importe à quel endroit de la ligne cette commande a été entrée, le curseur se place au début de la ligne. La commande est utile si l'on doit effectuer d'autres insertions dans une très longue ligne. CTRL P n'a pas une grande incidence dans les lignes courtes.

Le prochain groupe de commandes permet de positionner le curseur dans une ligne. En mode EDITOR, mis à part les touches "Flèches" BACKSPACE et RETYPE, il existe trois possibilités :

CTRL N : End

Le curseur est placé à la fin de la ligne.

CRTL B : Begin

Le curseur est placé sur le premier chiffre du numéro de ligne.

CTRL F : Find

Le curseur saute au caractère donné lorsque celui ci apparaît pour la première fois. Si on répète l'entrée du caractère le curseur se déplace jusqu'à ce que le caractère soit de nouveau rencontré.

Avec ce troisième groupe de commande il est possible de mémoriser les transformations d'une ligne ou de les supprimer.

CTRL R : Restart

Avec cette commande, toutes les modifications effectuées jusque là sont interrompues et la ligne réapparaît sous sa forme initiale. Ceci n'est possible que si on se trouve toujours sous Editor. Après CTRL R, on est toujours sous Editor,et on peut opérer d'autres modifications dans cette même ligne.

CTRL X : eXit

Toutes les modifications effectuées jusque là sont interrompues et on quitte EDITOR. La ligne qui vient d'être éditée est gardée en mémoire centrale sous sa forme initiale.
L'emplacement du curseur dans la ligne n'a pas d'importance.

CTRL Q : Quit

Avec cette commande la position du curseur est très importante. On est plus sous EDITOR, toutes les modifications introduites dans la ligne à gauche du curseur sont mémorisées tous les signes à droite y compris celui sous lequel se trouve le curseur, sont effacés. Cette commande permet de couper une partie de la ligne.

CTRL M : RETURN

Toutes les modifications dans la ligne sont mémorisées. On est en mode EDITOR, et la position du curseur lors de l'entrée n'a pas d'importance. On peut aussi appuyer simplement sur la touche RETURN au lieu de CTRL M.

On peut avec ces commandes effectuer toutes les modifications souhaitées. De plus on peut insérer dans une instruction PRINT, les commandes de curseur pour pouvoir se déplacer sur l'écran dans les 4 directions. En mode EDITOR on peut aussi passer en majuscules ou en minuscules avec la commande CTRL S. Si l'APPLE n'est pas pourvu du générateur de caractères correspondant, il y aura sur l'écran q'une suite de signes incompréhensibles; néanmoins on aura le texte en minuscules sur l'imprimante.

Pour illustrer les possibilités d'utilisation du programme EDITOR, voici un petit exemple: il s'agit d'un petit programme qui permet de calculer le prix de l'essence par personne dans une société de transport:

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.89 : REM PRIX DE L'ESSENCE EN F
40 INPUT KM
50 INPUT N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 40
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"DISTANCE EN KM   : ";KM
120 PRINT: PRINT"COUT PAR PERSONNE: ";KP
130 END
```

En ligne 30 la valeur P doit être modifiée car le prix de l'essence a baissé:

CTRL E	Le mot EDIT apparaît
30 RETURN	Le curseur est placé sur "P"
-- (4 fois)	Le curseur est placé sur "8".
3	Le chiffre "8" est remplacé par "7".
6	Le chiffre "9" est remplacé par "6".
RETURN	EDITOR est quitté, les modifications sont mémorisées:

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT KM
50 INPUT N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 40
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"DISTANCE EN KM   : ";KM
120 PRINT: PRINT"COUT PAR PERSONNE: ";KP
130 END
```

Dans la ligne 40 on désire ajouter une explication relative à la distance.

```
CTRL E          Appel de l'éditeur
40 RETURN       Le curseur est positionné sur le "I"
CTRL F K        Le curseur saute sur le "K" de "KM".
CTRL I          Mode insertion
"DISTANCE DU PARCOURS: "; Entrée du texte.
RETURN          Fin du mode insertion et des modifications.
```

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT"DISTANCE DU PARCOURS: ";KM
50 INPUT N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 40
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"DISTANCE EN KM     ";KM
120 PRINT: PRINT"COUT PAR PERSONNE: ";KP
130 END
```

On souhaite ajouter une explication en ligne 50 de la manière suivante:

```
CTRL E        Après l'appel d'EDITOR,
50 return     le curseur est placé sur le I.
--  (6 fois)  Le curseur est placé sur N, devant lequel se produit
              l'insertion.
CTRL I        Mode insertion.
"NOMBRE DE PERSONNES: "; Entrée du nouveau texte.
CTRL Q        On abandonne EDITOR, ce qui a pour effet de couper à
              la variable N.
```

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT"DISTANCE DU PARCOURS: ";KM
50 INPUT"NOMBRE DE PERSONNES: ";
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 40
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"DISTANCE EN KM   : ";KM
120 PRINT: PRINT"COUT PAR PERSONNE: ";KP
130 END
```

La ligne 50 doit être complétée.

```
CTRL E 50    RETURN
CTRL N       Le curseur saute derrière le ";"
N            Le "N" est replacé derrière le ";"
RETURN
```

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT"DISTANCE DU PARCOURS: ";KM
50 INPUT"NOMBRE DE PERSONNES: ";N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 40
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"DISTANCE EN KM   : ";KM
120 PRINT: PRINT"COUT PAR PERSONNE: ";KP
130 END
```

En ligne 60, il y a une fausse adresse de branchement, il faut remplacer le numéro de ligne 40 par 50.

```
CTRL E 60    RETURN
CTRL F 4     Le curseur saute tous les signes jusqu'au 4
5            Entrée du chiffre 5
RETURN
```

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT"DISTANCE DU PARCOURS: ";KM
50 INPUT"NOMBRE DE PERSONNES: ";N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 50
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"DISTANCE EN KM   : ";KM
120 PRINT: PRINT"COUT PAR PERSONNE: ";KP
130 END
```

La division par 100 en ligne 70 est une erreur, il faut la supprimer:

```
CTRL E 70    RETURN
-- (10 fois) Le curseur est placé sur le 1
CTRL D       Le 1 est effacé
CTRL D       Le 0 est effacé
CTRL D       Le 0 est effacé
```

Voici maintenant le programme:

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT"DISTANCE DU PARCOURS: ";KM
50 INPUT"NOMBRE DE PERSONNES: ";N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 50
70 GV=KM/*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"DISTANCE EN KM   : ";KM
120 PRINT: PRINT"COUT PAR PERSONNE: ";KP
130 END
```

On s'est trompé. La ligne était correcte.

CTRL X On abandonne l'éditeur, les modifications sont supprimées, la ligne reste dans sa version initiale.

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT"DISTANCE DU PARCOURS: ";KM
50 INPUT"NOMBRE DE PERSONNES: ";N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 50
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"DISTANCE EN KM   : ";KM
120 PRINT: PRINT"COUT PAR PERSONNE: ";KP
130 END
```

On désire imprimer le nombre de personnes et conserver la numérotation des lignes par pas de dix, on rebaptise donc les lignes 110-130 en 120-140.

```
CTRLE 130    RETURN
CTRL B       Le curseur saute sur le premier chiffre du numéro de
             ligne.
--  4        Le 3 est changé en 4
RETURN
CTRL E 120   RETURN
CTRL B
--  3        2 devient 3
RETURN
CTRL E 110   RETURN
CTRL B
--  2        1 devient 2
RETURN
```

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT"DISTANCE DU PARCOURS: ";KM
50 INPUT"NOMBRE DE PERSONNES: ";N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 50
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"DISTANCE EN KM   : ";KM
120 PRINT: PRINT"DISTANCE EN KM   : ";KM
130 PRINT: PRINT"COUT PAR PERSONNE: ";KP
140 END
```

Les lignes 110 et 120 sont maintenant identiques. La ligne 110 va avoir un nouveau texte. On commence par effacer le premier:

CTRL E 110 RETURN
CTRL D (7 fois)

Après cela on modifie la chaîne. Voici le programme:

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT"DISTANCE DU PARCOURS: ";KM
50 INPUT"NOMBRE DE PERSONNES: ";N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 50
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT"DISTANCE EN KM   : ";KM
120 PRINT: PRINT"DISTANCE EN KM   : ";KM
130 PRINT: PRINT"COUT PAR PERSONNE: ";KP
140 END
```

On remarque l'erreur avant de quitter EDITOR. Le début de ligne peut être récupéré par:

```
CTRL R      Les modifications sont suspendues mais l'édition peut
            continuer
CTRL F S    Le curseur est placé sur le D de DISTANCE.
"NOMBRE DE PERSONNES" L'ancienne chaîne est écrite.
RETURN
```

```
3 HOME : HTAB(9) : PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984" :PRINT
10 REM PROGRAMME POUR LE CALCUL DU COUT D'UN DEPLACEMENT EN AUTO EN
FONCTION DU NOMBRE DE PERSONNES ET DES DISTANCES
15 REM LE PRIX DE L'ESSENCE ET LA CONSOMMATION SONT FIXES
20 V=10 : REM CONSOMMATION EN L POUR 100 KM
30 P=5.76 : REM PRIX DE L'ESSENCE EN F
40 INPUT"DISTANCE DU PARCOURS: ";KM
50 INPUT"NOMBRE DE PERSONNES: ";N
60 PRINT: IF N 5 THEN PRINT"PLUS DE 5 PERSONNES DANS UNE VOITURE, C'EST
DEFENDU !": PRINT: GOTO 50
70 GV=KM/100*V
80 K=GV*P/100
90 KP=K/N: KP=INT(KP*100+.5)/100
100 HOME
110 PRINT: PRINT"NOMBRE DE PERSONNES: ";KM
120 PRINT: PRINT"DISTANCE EN KM   : ";KM
130 PRINT: PRINT"COUT PAR PERSONNE: ";KP
140 END
```

Le programme doit avoir ainsi trouvé sa forme définitive et on ne désire pas d'autres modifications. Dans cet exemple toutes les instructions n'ont pas été décrites mais il est clair qu'un programme tel que LINE EDITOR permet des modifications aisées.
Le regroupement de lignes peut s'effectuer comme en mode ESCAPE, on peut cependant éviter POKE 33,33 avec CTRL E.

On a déjà vu qu'avec PLE, certaines commande ESC avaient une signification particulière. Ainsi par exemple, ESC : commande l'appel du moniteur.

Un inconvénient de PLE réside dans sa manière compliquée de numéroter les lignes. L'utilisation du programme amène pourtant un certain avantage, les lignes ne devant plus être relues avec RETYPE.

Dans le chapitre suivant un programme d'aide ,qui dans ce domaine est bien plus confortable, sera mis sur pied.

Chapitre 7: Aides à la programmation

7.1 Remarques préliminaires

Sous le génériques d'aides à la programmation nous allons parler de plusieurs techniques qui rendent un programme plus rapide ou plus agréable à utiliser grâce à des routines d'aide qui faciliteront le travail de programmation.

Dans les deux premières parties nous présenterons quelques possibilités mises en oeuvre par les packages APPLE DOC et DOS TOOL KIT, puis les routines APA (Apple Programmer's Assistant), LINE DOC et VARDOC qui sont des aides indispensables pour l'utilisateur d'APPLE qui écrit des programmes importants dans lesquels la vue d'ensemble se perd.

Dans la troisième partie, il sera démontré par l'exemple comment on peut mettre à sa disposition dès la mise en route de l'ordinateur les routines les plus fréquemment utilisées.

Les deux dernières parties présente une programmation avancée des deux problèmes déjà rencontrés par l'utilisateur: le tri et la présentation de l'écran.

Avec le problème du classement, il montre plus particulièrement comment le choix d'un algorythme approprié peut accélérer le déroulement d'un programme.

La formation de masques d'écran bien disposés nécessite de longues heures de travail.

Grâce à de petits trucs on pourra économiser un temps appréciable.

7.2 Numérotation et fusion de programmes

Le DOS TOOL KIT (Tool Kit=boîte à outils) avec copyright APPLE COMPUTER CORPORATION comprend mis à part beaucoup d'autres programmes (jeux...) APA, une aide pour les programmes plus longs avec lequel des programmes entiers ou des parties de programmes peuvent être numérotés ou mixés.
Les programmes APA et LOADAPA se trouve sur la disquette. Le programme doit être lancé par

```
RUN LOADAPA
```

celui-ci se charge sous HIMEM puis HIMEM est réduit de manière à protéger le programme contre tout écrasement.

La réduction de HIMEM permet donc la protection de APA et autorise l'utilisateur à travailler normalement, les programmes pouvant être chargés et sauvés comme si APA n'existait pas.
Lorsque la transformation se produit, la mémoire libre et disponible est réduite avec HIMEM.
Après le bootage, HIMEM pointe sur 38 400, après le chargement d'APA l'adresse maximale utilisable est 34 785 .
Cette perte de 3615 octets n'est sensible qu'avec des programmes très importants contenant de nombreuses variables.
LOADAPA envoie l'impression de l'ensemble des commandes APA disponibles.

```
APPLESOFT PROGRAMMER'S ASSISTANT
VERSION 1.0
(C) COPYRIGHT 1979, APPLE COMPUTER INC.

&RENUMBER  START , INC , FIRST , LAST
&HOLD
&MERGE
```

```
&LENGTH
&COMPRESS
&SHOW
&NOSHOW
&AUTO  START , INC
&MANUAL
&XREF
&KEYS
```

Comme nous l'avons déjà expliqué dans le résumé, toutes les commandes APA doivent être remarquées par le signe "&" pour les différencier des commandes usuelles en mode direct.
Les commandes &RENUMBER,&HOLD et &MERGE sont utilisées dans des cas particuliers que nous détaillerons.

&LENGTH

Donne la longueur d'un programme en mémoire centrale. Après la composition ou le chargement d'un programme, il faut taper.

&L

et la longueur du programme est renvoyée en octets,(En décimal et hexadécimal.)

&COMPRESS

Tous les REM sont supprimés du programme après la frappe de

&C

Suit une impression donnant le nombre d'octets économisés.(En décimal et en hexadécimal).

&SHOW

Après l'entrée de la commande

&S

les caractères de commande qui n'étaient pas apparents, le deviennent et ce en écriture inverse. La commande reste en fonction jusqu'à ce qu'elle soit supprimée avec la commande

&NOSHOW

&N

supprime la commande &S et les caractères de commande ne sont plus apparents.

&AUTO START , INC

Cette commande permet de numéroter des lignes, en commençant à la ligne START et en respectant un pas de progression INC. Les deux paramètres doivent être donnés avec la commande. Par exemple:

& A 100,10

La numérotation automatique débute avec 100 pour numéro de première ligne, puis 120, 130...

On peut effectuer une numérotation "manuelle", lorsque l'on désire s'écarter du pas de progression INC . Si dans l'exemple le numéro de ligne 120 n'est pas généré automatiquement mais en tapant les chiffres, il ne sera pas enregistré par APA, le numéro est établi automatiquement.

Si on tape aussitôt sur return , cette ligne ne sera pas dans le listing, mais APA suspend le pas de progression correspondant. C'est seulement avant la fin de la ligne, avant return , en entrant CTRL X, la suspension de la numérotation des lignes se fait. On l'utilise également pour ressortir des lignes dont les débuts sont faux.On peut transformer le pas de progression avec d'autres commandes, même si une partie du programme a déjà été entrée. On peut suspendre cette commande avec:

&MANUAL

&MA

la numérotation automatique des lignes est stoppée.

&XPRET

Cette commande établit une liste alphabétique de toutes les variables utilisées dans un programme. Les variables dimensionnées sont mises en évidence par des parenthèses gauches, les variables de chaînes et entières

sont indiquées comme d'habitude avec: $ et %. On peut interrompre l'impression avec CTRL S. On tape simplement:

&X

Nous donnerons dans la prochaine partie d'autres explications pour les tableaux de variables.

&KEYS

Avec cette commande on peut disposer de 3 caractères supplémentaires qui peuvent être appelés ensuite par des caractères de commande. Après avoir tapé

&K

le crochet gauche ( ) correspond à CTRL K, le backslash (/) à CTRL L et le trait sous la ligne à CTRL O. Cette fonction est suspendue avec

&MA

Avant de parler des trois commandes les plus importantes d'APA, il faut préciser que certaines de ces commandes comme par exemple & AUTO perturbent d'autres routines d'aide, si celles-ci ont été chargées auparavant. Nous reparlerons de ce problème dans le paragraphe 7.4.
Maintenant voici trois commandes dont nous allons expliquer la syntaxe:

&HOLD

Le programme est écarté dans la mémoire centrale, il est possible de charger un nouveau programme, l'ancien n'est pas perdu.

&MERGE

Deux programmes peuvent être fusionnés de la manière suivante:
Le premier programme est chargé avec

&H

puis le second avec

&M

ses numéros de lignes doivent être différents de ceux du premier. Les deux programmes sont mélangés selon leurs numéros de lignes.
Si des numéros sont en double, on voit apparaître

CONTINUE ?

On interrompt la suite avec "N", avec "Y" l'une des lignes est enlevée du programme, l'autre est conservée.

&RENUMBER, START , INC , FIRST , LAST

Avec cette commande, la numérotation de tout ou une partie du programme peut être modifiée, si il ne s'est pas produit de chevauchement de numéros de lignes. Si cela s'est produit, la commande est abandonnée et un message l'indique.
Les quatre paramètres ne sont pas obligatoires, la commande a néanmoins besoin des virgules. Si il manque l'un des paramètres dans l'énoncé, les valeurs par défaut ne sont pas prises en compte. Exemple:

START : 100 On prévoit 100 comme premier numéro de ligne
INC : 10 La numérotation des numéros de lignes se fait par 10 (c'est-à-dire par pas de progression égal à 10).
FIRST : 0 Le plus petit numéro de ligne possible est le 0 donc la renumérotation commence à la première ligne du programme.
LAST :63999 C'est le numéro de ligne le plus grand donc la modification sera pratiquée jusqu'à la dernière ligne du programme.

Voici quelques explications supplémentaires. Ce programme est en mémoire centrale:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 REM DEMONSTRATION DE
11 REM     RENUMBER
12 REM
20 PRINT"R"
30 PRINT"E"
40 PRINT"N"
50 PRINT"U"
60 PRINT"M"
70 PRINT"B"
80 PRINT"E"
90 PRINT"R"
100 END
```

Si on frappe

```
          &R 100,,10
```

on obtient la version suivante:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
100 REM DEMONSTRATION DE
110 REM     RENUMBER
120 REM
130 PRINT"R"
140 PRINT"E"
150 PRINT"N"
160 PRINT"U"
170 PRINT"M"
180 PRINT"B"
190 PRINT"E"
200 PRINT"R"
210 END
```

Pour remettre en place la version initiale, il faut utiliser deux pas:

```
&R 10, 1, 100, 120
```

Le premier nouveau numéro de ligne est 10, le pas de progression doit être 1 de la ligne 100 à 120.
Voici le résultat:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 REM DEMONSTRATION DE
11 REM     RENUMBER
12 REM
130 PRINT"R"
140 PRINT"E"
150 PRINT"N"
160 PRINT"U"
170 PRINT"M"
180 PRINT"B"
190 PRINT"E"
200 PRINT"R"
210 END
```

La commande RENUMBER remet en place la numérotation initiale du programme avec:

```
&R20,,130
```

La premiere ligne nouvelle est 20, la valeur par défaut du pas est 10, la numérotation commence à la ligne 130 du programme et continuera jusqu'à la fin du programme car aucune valeur de fin n'a été précisée. Maintenant la première version se trouve de nouveau en mémoire. La commande suivante n'est pas appelée par APA mais elle produit un message d'erreur:

```
INTERLEAVED OR DUPLICATE line number
```

```
&R 36,6,70,90.
```

L'ordinateur utilise les lignes 70,80,90, du programme ce qui donne l'affichage B,E,et R pour numéroter avec une progression de 6 en commençant par 36, on obtient les numéros de lignes suivants:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 REM DEMONSTRATION DE
11 REM     RENUMBER
12 REM
20 PRINT"R"
30 PRINT"E"
36 PRINT"B"
40 PRINT"N"
42 PRINT"E"
48 PRINT"R"
50 PRINT"U"
60 PRINT"M"
100 END
```

Cette numérotation n'est pas établie avec APA, mais avec RENUMBER du système de disquette APPLE.

On voit que les nouveaux et les anciens numéros de lignes "se chevauchent", car la nouvelle ligne 36 est classée entre 30 et 40, les deux autres (42 et 48) derrière la ligne 40. On ne développe pas ces insertions mais il est possible d'intercaler une ou plusieurs lignes entre 2 lignes d'un programme avec la commande:

```
&R 53,3,70,90.
```

On a maintenant cette numérotation :

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 REM DEMONSTRATION DE
11 REM     RENUMBER
12 REM
20 PRINT"R"
30 PRINT"E"
40 PRINT"N"
50 PRINT"U"
53 PRINT"B"
56 PRINT"E"
59 PRINT"R"
60 PRINT"M"
100 END
```

Les décalages dans le programme deviennent nécessaires lorsque le déroulement du programme n'est pas satisfaisant. Après avoir transformé le déroulement du programme, on peut mettre en place de nouveaux pas de progression avec un autre RENUMBER. L'exemple suivant ne peut pas être suivi car ses numéros de lignes seraient en double.

```
&R 10,10,10,12
```

Les lignes 10, 11 et 12 doivent être déplacée en 10, 20 ,30 d'où une double utilisation des lignes 20 et 30.
L'exécution n'a pas lieu et le message apparaît:

```
INTERLEAVED OR DUPLICATE LINE NUMBER
```

Cette liste des possibilités met en évidence les avantages de la routine RENUMBER.
La numérotation ne concerne pas uniquement l'établissement des numéros de

lignes mais elle s'occupe aussi des adresses de branchement utilisées dans les instructions GOTO, IF...THEN, GOSUB.

Pour chaque numéro de ligne modifié, on vérifie si il est appelé par une instruction de branchement et, le cas échéant, l'adresse est modifiée.
La vérification porte sur l'ensemble du programme, même si une partie de celui-ci est renumérotée.
Terminons cette partie avec un exemple illustrant l'utilité d'APA.
Même si la disquette APA comporte un programme RENUMBER, le programme APA permet de travailler avec EDITOR avec lequel 3 des commandes les plus importantes sont compatibles.

Il y a parfois des interactions avec EDITOR qui conduisent à l'appel du moniteur après un RENUMBER.
La commande est pourtant accomplie mais il faudra taper 3DOG pour quitter le moniteur.

Le petit programme d'affichage suivant doit être inséré comme sous-programme dans un programme plus vaste. Les variables H1 et A$ sont transmise par le programme principal

```
10      H$="                                                ":      A=LEN(A$):
A$=A$+RIGHT$(H$,40-A)
20 H$=LEFT$(A$,1): A$=RIGHT$(A$,39) +H$: INVERSE: VTAB(H1) :CALL -958
30 VTAB(H1): PRINT LEFT$(A$,40): FOR Q=1 TO 150: NEXT
40 IF PEEK(-16384) 128 THEN 20
50 NORMAL
```

On désire mettre une en-tête de ce type au programme:

```
30 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
40 PRINT: PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984"
```

Le programme principal permet de calculer la moyenne des valeurs N. Il s'est glissé une erreur que le programmeur a remarqué à temps.

```
10 HTAB 7: PRINT"CALCUL DE LA MOYENNE"
20 A$="APPUYEZ SUR UNE TOUCHE": H1=20: GOSUB 100
30 INPUT"VALEUR";X
40 INPUT"NOMBRE DE VALEURS";N
50 S=S+X: I=I+1: IF I N THEN 30
60 HOME : VTAB10: PRINT"MOYENNE DES ";N;" VALEURS: ";S/N
70 END
```

Il s'agit de permuter les lignes 30 et 40. Il faut mettre en place le programme en-tête avant la ligne 10. A partir de la ligne 100, le sous-programme d'affichage permet l'impression du message de la ligne 20. Les programmes d'affichage et d'en-tête se trouvent sur la disquette; dans la mémoire centrale, le programme de calcul de la moyenne avec l'erreur. Que va-t-il se passer maintenant?

```
1. &R 25,,40,40
```

Avec cette commande la ligne 40 devient la ligne 25. Le déroulement du programme principal est correct:

```
10 HTAB 7: PRINT"CALCUL DE LA MOYENNE"
20 A$="APPUYEZ SUR UNE TOUCHE": H1=20: GOSUB 100
25 INPUT"NOMBRE DE VALEURS";N
30 INPUT"VALEUR";X
50 S=S+X: I=I+1: IF I N THEN 30
60 HOME : VTAB10: PRINT"MOYENNE DES ";N;" VALEURS: ";S/N
70 END
```

```
&H
```

Le programme principal est mis de côté.

2. LOAD EN-TETE

Le premier programme est chargé. Il comporte les numéros de lignes 30 et 40 qui vont être transformés en 3 et 4:

```
&R 3, 1
```

Après cela 2 nouvelles lignes sont insérées dans le programme principal avec:

```
&M
```

On obtient alors le programme suivant:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984"
10 HTAB 7: PRINT"CALCUL DE LA MOYENNE"
20 A$="APPUYEZ SUR UNE TOUCHE": H1=20: GOSUB 100
25 INPUT"NOMBRE DE VALEURS";N
30 INPUT"VALEUR";X
50 S=S+X: I=I+1: IF I N THEN 30
60 HOME : VTAB10: PRINT"MOYENNE DES ";N;" VALEURS: ";S/N
70 END
```

Ceci est mis de côté avec &H.

3. Il manque encore le sous-programme, on le charge avec

```
LOAD AFFICHAGE.
```

On insère une ligne 60 avec un RETURN pour le transformer en sous-programme

et on efface l'écran avant:

```
60 HOME: RETURN
```

La commande GOSUB 100 étant dans le programme principal, il faut le numéroter:

```
&R
```

suffit pour transférer les lignes 10 à 60 dans les lignes 100/150. L'adresse de branchement située dans l'ancienne ligne 40 (maintenant 130) est transformée en 110. La fusion des programmes s'effectue avec

```
&M
```

Enfin

```
& 30,,25
```

pour éviter le numéro de ligne 25 qui n'est pas habituel.
Le programme est terminé:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984"
10 HTAB 7: PRINT"CALCUL DE LA MOYENNE"
20 A$="APPUYEZ SUR UNE TOUCHE": H1=20: GOSUB 100
30 INPUT"NOMBRE DE VALEURS";N
40 INPUT"VALEUR";X
50 S=S+X: I=I+1: IF I N THEN 40
60 HOME : VTAB10: PRINT"MOYENNE DES ";N;" VALEURS: ";S/N
70 END
100      H$="                                          ":      A=LEN(A$):
```

```
A$=A$+RIGHT$(H$,40-A)
110 H$=LEFT$(A$,1): A$=RIGHT$(A$,39) +H$: INVERSE: VTAB(H1) :CALL -958
120 VTAB(H1): PRINT LEFT$(A$,40): FOR Q=1 TO 150: NEXT
130 IF PEEK(-16384) 128 THEN 110
140 NORMAL
150 HOME: RETURN
```

Pendant la numérotation, les adresses de branchement sont également modifiées. Si on désire insérer plus de 2 lignes numérotées par pas de dix dans le programme principal, la numérotation du sous-programme doit être plus grande, avec des lignes par exemple commençant par 200. La commande

```
&R 200,,100
```

permet une numérotation qui transfère l'adresse de branchement à l'intérieur du sous-proogramme de 110 à 210, et établit l'adresse de branchemennt derrière GOSUB placé à 200 dans le programme principal.

Les deux routines d'aide APA et PROGRAMM LINE EDITOR, sont toutà fait compatibles l'une avec l'auutre, on charge d'abord EDITOR puis APA. APA est perturbé par FDITOR lorsqu'il y a une suite de séries inverses. Pour fusionner des parties de programmes déterminées, l'utilisation de fichiers de texte et de la commande EXEC, comme nous l'avions décrite dans le chapitre 5, n'est pas aussi pratique que la mise en oeuvre d'APA dont les possibilités sont variées.

7.3. Vue d'ensemble et documentation.

Lors de l'élaboration d'un programme important, ou de la fusion de programmes, le problème du contrôle des variables se pose, il faut savoir quels noms de variables utiliser pour éviter toute coïncidence non désirée entre les différentes parties de programme. Il est parfois utile, avec de très gros programmes, de réduire le nombre de variables, c'est-à-dire d'utiliser plusieurs fois le même nom pour des variables différentes: ces variables n'étant utilisées qu'une fois.

On perd très rapidement la vue d'ensemble sur les possibilités d'un programme. C'est pourquoi on le documente avec l'instruction REM, sauf si le manque de mémoire se fait sentir.
Si on désire modifier des numéros de lignes, par exemple pour regrouper plusieurs lignes, il faut penser à modifier les branchements sur ces lignes.

Un progiciel donnant une vue d'ensemble des variables utilisées ainsi que les adresses de branchement et permettant d'établir un programme avec documentation externe, s'avère, alors, fort utile.
Ce progiciel s'appelle APPLE DOC avec copyright de Roger Wagner et de la firme Southwestern DATA systems, Californie, USA. Il est constitué de 3 parties: VARDOC, LINE DOC,et REPLACE. Avec la commande EXEC+Nom on peut faire démarrer 3 fichiers textes. EXEC permet la consolidation du programme en cours de traitement.
Les emplacements mémoire 103 et 104 contiennent les adresses de début de programme; 175 et 176, la fin de programme. Si le contenu de 103 et 104 est supérieure à l'adresse de fin de programme, le programme initial est sauvegardé. Maintenant le programme peut être chargé. Un des 3 programmes APPLESOFT VRF, LRF et RPL est chargé et démarré par l'un des 3 fichiers EXC. A la fin de ces opérations, on retrouve les anciennes valeurs des 4 emplacements mémoire car ils ont été mémorisés en 1912-1915 grâce aux instructions des fichiers EXEC.

7.3.1 Contrôle des variables.

Une nouvelle affectation de l'emplacement de mémoire se produit après EXEC VARDOC, ainsi que la confirmation de la valeur de départ et de début du programme APPLESOFT VRF, qui provoque l'apparition du titre, du copyright, d'explications et demande quel est le titre du programme recherché.
Le début et la fin du programme sont établis, ensuite on demande le nom du programme recherché. Le tableau recherché est mis en place et il est protégé par la réduction de HIMEM. Lorsque l'imprimante est mise en route, on obtient la présentation du résultat: une liste de tous les noms des variables et les numéros de lignes dans lesquelles elles interviennent. Dans l'exemple suivant on a un tableau des variables du programme du paragraphe 2 pour le calcul des moyennes.

```
*******************************
*                             *
*          DEMO VARDOC        *
*                             *
*   -- TABLE OF VARIABLES --  *
*                             *
*******************************
```

A
100 100

A$
20 100 100 100 110 110 110 120

H$
100 100 110 110

H1
20 110 120

I
50 50 50

N
30 50 60

Q
120

S

```
50 50 60

X
40 50

END OF VAR. LIST
```

Ensuite le programme retourne à son menu:

```
DO YOU WANT TO:(1) RETURN
               (2) REVIEW LIST AGAIN
               (3) NEW ANALYSIS
               (4) ENTER DESCRIPTORS
               (5) SAVE TO DISK
               (6) GET FROM DISK
```

L'option 1 permet le retour au programme en cours de traitement, donc la fin de VARDOC, et le rétablissement des adresses 103, 104, 175 et 176 à leurs valeurs initiales, HIMEM n'est plus abaissé et les adresse 115, 116 sont modifiées.

L'option 2 propose 3 possibilités:
L'entrée de 2 provoque les mêmes messages que le 1 mais avec la possibilité d'une sortie sur imprimante. L'entrée de '2*' prooduit une liste réduite qui ne contient que les noms des variables sans numéros de lignes correspondants.

```
*******************************
*                             *
*          DEMO VARDOC        *
*                             *
*   -- TABLE OF VARIABLES --  *
*                             *
*******************************
```

```
A A$ H$ H1 I N Q S X

END OF VAR. LIST
```

On peut également entrer '2**' mais nous ne l'expliqueroons pas ici.

L'option 3 efface l'ancienne liste de variables, et il est possible d'analyser le programme.

Les options 4-6 se rapportent à la documentation d'un programme. L'option 4 permet l'entrée d'explications courtes pour les noms des variables. Dans ce mode, RETURN signifie non pas une entrée mais que l'ancien contenu de la description est conservé. On appelle la liste des numéros de lignes des variables avec L. Avec '/' on efface les désignations anciennes qui n'ont pas de correspondance et elles sont remplacées par de nouvelles entrées. 'X' permet d'arrêter une entrée pour y ajouter la fin de la liste de variables.

Avec l'option 5 les désignations de variables sont inscrites sur la disquette en fichier texte, avec un nom composé du nom du programme et d'un VFILE supplémentaire.

On peut lire le fichier texte correspondant avec l'option 6.
Après l'appel de l'option 4 le programme retourne au menu.
Les 3 possibilités de l'option 2, ont été utilisées, voici le résultat:

```
*********************************
*                               *
*          DEMO VARDOC          *
*                               *
*   -- TABLE OF VARIABLES --    *
*                               *
*********************************
```

A - LONGUEUR DU TEXTE
100 100

A$- TEXTE EN SORTIE
20 100 100 100 110 110 110 120

H$- CHAINE DE 40 ESPACES
100 100 110 110

H1- NUMERO DE LA CHAINE SORTIE
20 110 120

I - COMPTEUR POUR ENTREE D'UNE VALEUR
50 50 50

N - NOMBRE DE VALEURS
30 50 60 60

```
Q - INDEX DE BOUCLE
120

S - SOMME DES VALEURS
50 50 60

X - VALEUR
40 50

END OF VAR. LIST
```

"2*": même effet que précédemment

"2**":

```
*********************************
*                               *
*          DEMO VARDOC          *
*                               *
*   -- TABLE OF VARIABLES --    *
*                               *
*********************************
```

A - LONGUEUR DU TEXTE

A$- TEXTE EN SORTIE

H$- CHAINE DE 40 ESPACES

H1- NUMERO DE LA CHAINE SORTIE

I - COMPTEUR POUR ENTREE D'UNE VALEUR

N - NOMBRE DE VALEURS

X - VALEUR

END OF VAR. LIST

7.3.2. Contrôle des adresses de branchement.

Il se passe la même chose avec EXEC LINE DOC qu'avec VARDOC: Protection du programme, chargement et démarrage du programme Applesoft LRF, qui produit un titre puis demande le paragraphe à rechercher, le nom du programme, et si l''imprimante doit être mise en route.

Une liste de tous les numéros de lignes apparaissant comme adresses de branchements est établie pour chaque ligne où se trouve la commande de branchement correspondante. Pour l'exemple du calcul de la moyenne que nous connaissons déjâ, nous avons le tableau suivant:

```
**********************************
*                                *
*          DEMO LINEDOC          *
*                                *
*   -- TABLE OF LN  XREFS --     *
*                                *
**********************************
```

```
     60
50

     100
20

     110
130

END OF LN  LIST
```

Sur l'écran les adresses de branchement (60, 100, 110) sont décalées et écrites en inverse, pour les distinguer des autres. Le menu de LINE DOC est à peu près équivalent à celui de VARDOC, les options sont les suivantes:

```
DO YOU WANT TO: (1) RETURN TO PROGRAM
                (2) REVIEW LIST AGAIN
                (3) NEW ANALYSIS
                (4) ENTER DESCRIPTORS
                (5) SAVE TO DISK
                (6) GET FROM DISK
```

Dans l'option 2 il y a également les 3 possibilités dont nous avons parlé. Après le rappel de l'option 4 on a le message suivant:

```
"2" :
          ********************************
          *                              *
          *          DEMO LINEDOC        *
          *                              *
          *   -- TABLE OF LN  XREFS --   *
          *                              *
          ********************************

    60 - ENTREE DES DONNEES
50

    100- SOUS-PROGRAMME LIGNE
20

    110- BOUCLE POUR IMPRESSION
130

END OF LN  LIST
```

```
"2*"  :
          ********************************
          *                              *
          *          DEMO LINEDOC        *
          *                              *
          *   -- TABLE OF LN  XREFS --   *
          *                              *
          ********************************

   40  100 110

END OF LN  LIST

"2**"  :

          ********************************
          *                              *
          *          DEMO LINEDOC        *
          *                              *
          *   -- TABLE OF LN  XREFS --   *
          *                              *
          ********************************



40 - ENTREE DES DONNEES

100- SOUS-PROGRAMME LIGNE

110- BOUCLE POUR IMPRESSION

END OF LN  LIST
```

7.3.3 Modifications

L'utilisation de VARDOC et LINE DOC met en évidence la nécessité de modifications à l'intérieur d'un programme. On met en place maintenant le troisième programme du progiciel APPLE DOC. Il est initialisé avec EXEC REPLACE. Avant de commencer, le programme initial est protégé par la nouvelle affectation des adresses 103 ,104. Pour cela on aura besoin d'une mémoire tampon supplémentaire qui peut aller jusqu'à 256 octets. C'est nécessaire car le programme s'allonge avec les modifications: exemple la variable A peut s'appelle maintenant A1. On notera une autre différence dans le déroulement du programme: après le titre c'est le menu qui apparaît en premier, il comporte 6 options:

```
DO YOU WANT:
   (1) REPLACE A VARIABLE/
   (2) REPLACE A LITERAL
   (3) TRUNCATE & REPLACE A LITERAL
   (4) DOCUMENT A LITERAL
   (5) LIST A LINE
   (6) RETURN PROGRAM
```

L'option 6 permet comme dans l'option 1 des 2 menus précédents , la "libération " du programme initial.

L'option 5 permet de lister des parties de programme lorsqu'on a pas encore de modifications à effectuer.

Dans l'option 1 on peut modifier des noms de variable ou des adresses de branchements. Apres l'entrée de "1" on précise d'abord les zones de programme intéressées.

Les 2 entrées suivantes concernent le nom de la variable ou l'adresse de

branchement, de leurs anciennes et nouvelles valeurs. Puis on précise si l'ancienne valeur doit être modifiée dans tout le programme ou seulement dans quelques lignes isolées:

```
WHAT CHAR'S TO REPLACE? A$
WHAT CHAR'S TO PUT THERE? B$
ALL OR SOME? (A/S):A
```

Le programme de calcul de la moyenne est modifié de la manière suivante:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9) :PRINT"R.PRUST / W.VOSS 1984": PRINT
10 HTAB 7 :PRINT"CALCUL DE LA MOYENNE"
20 B$="APPUYEZ SUR UNE TOUCHE": H1=20: GOSUB 100
30 INPUT"NOMBRE DE VALEURS ";N
40 INPUT"VALEUR ";X
50 S=S+X: I=I+1: IF I N THEN 40
60 HOME: VTAB 10: PRINT"MOYENNE DES";N;"VALEURS: ";S/N
70 END
100 H$="                                          ": A=LEN(B$):
   B$=B$+RIGHT$(H$,40-A)
110 H$=LEFT$(B$,1): B$=RIGHT$(B$,39)+H$: INVERSE: VTAB(H1): CALL-958
120 VTAB(H1): PRINT LEFT$(B$,40): FOR Q=1 TO 150: NEXT
130 IF PEEK(-16384) 128 THEN 110
140 NORMAL
150 HOME: RETURN
```

La variable A$ est remplacée par B$.

On répond par "S" à la troisième question "ALL OR SOME", l'utilisateur est consulté pour chaque opération et il peut modifier le programme comme il le désire. Ce procédé est utile surtout si il se produit une double appellation comme cela peut arriver lors de la fusion de 2 programmes différents.

Les noms des tableaux doivent être précisés d'une parenthèse gauche. Dans l'option 1, à la place d'une variable numérique on peut donner une adresse de branchement qui servira pour le regroupement de lignes ou la supression de REM.

Avec l'option 2 on peut rechercher ou établir des chaînes dans un programme. Apres les questions concernant les zones de programme intéressées il faut répondre à : "ALL OR SOME". Ensuite, entrer après le

numéro de ligne 740 l'ancienne chaîne, derrière le numéro de ligne 750, la nouvelle. Le déroulement du programme est interrompu jusqu'à ce que l'on tape la commande

GOTO 760.

Comme exemple on entre en 740 MOYENNE et en 750 VALEUR MOYENNE. On obtient la chaîne suivante avec l'option ALL (les lignes 30 et 80 sont modifiées):

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9) :PRINT"R.PRUST / W.VOSS 1984": PRINT
10 HTAB 7 :PRINT"CALCUL DE LA MOYENNE"
20 B$="APPUYEZ SUR UNE TOUCHE": H1=20: GOSUB 100
30 INPUT"NOMBRE DE VALEURS ";N
40 INPUT"VALEUR ";X
50 S=S+X: I=I+1: IF I N THEN 40
60 HOME: VTAB 10: PRINT"VALEUR MOYENNE DES ";N;"VALEURS: ";S/N
70 END
100 H$="                                        ": A=LEN(B$):
   B$=B$+RIGHT$(H$,40-A)
110 H$=LEFT$(B$,1): B$=RIGHT$(B$,39)+H$: INVERSE: VTAB(H1): CALL-958
120 VTAB(H1): PRINT LEFT$(B$,40): FOR Q=1 TO 150: NEXT
130 IF PEEK(-16384) 128 THEN 110
140 NORMAL
150 HOME: RETURN
```

L'option 3 est utilisée pour supprimer les lignes REM lorsque dans un programme important, l'emplacement de mémoire est devenu trop juste: on ne remplace pas uniquement la chaîne dans la ligne 740 mais aussi tout ce qui se trouve à sa droite.
L'option est automatiquement 'SOME':
Exemple: 740 REM
750 .

Il est possible de remplacer "REM" par ":", et également de "couper" (truncated) le commentaire situé à la droite de REM.

Avec l'option 4, il est possible de dresser une liste de toutes les lignes dans lesquelles une chaîne déterminée apparaît avant d'effectuer une quelconque modification. Il est poossible également de mettre en route l'imprimante.

Nous avons donc vu les possibilités du progiciel APPLEDOC. L'utilisation de ces routines d'aide ne pose des problèmes qu'avec de très longs programmes, car les routines et les tableaux occupe de la place mémoire. Le programme a besoin de 6K octets; pour le tableau, cela dépend de sa taille et de chaque étude de programme. On peut résoudre ce problème de place de 2 manières différentes:

1- En raccourcissant le programme d'aide (environ 3.3 par VRF)

2- En effaçant dans la mémoire centrale des parties du programme initial qui pourront être analysées en plusieurs fois.

7.4 Initialisation.

Dans le chapitre 2 on a déjà expliqué rapidement que chaque nouvelle disquette doit être initialisée avec la commande

INIT Nom

Ceci se passe en trois étapes:

1- Tout d'abord la disquette est formatée, c'est-à-dire qu'elle est divisée en pistes (ce sont des anneaux concentriques autour de l'orifice central appelés tracks) et en secteurs (ce sont les parties d'un anneau). Avec APPLE II dispose de 35 pistes comprenant 16 secteurs, chaque secteur comporte 256 octets. Les pistes sont numérotées en hexadécimal de l'extérieuur vers l'intérieur ($0 à $22), les secteurs de $0 à $F.

2- le DOS est écrit sur la disquette dans les tracks $0 à $2. La piste $ 11 est aménagée pour la table des matières de la disquette.

3- Le programme de bienvenue est mémorisé sous le nom donné après INIT.

Après initialisation il reste 31 pistes de 16 secteurs comportant 256 octets, soit 124 K déduction faite de la place occupée le programme de bienvenue. Ce programme ne peut contenir qu'une ligne en BASIC, avec une seule instruction. C'est pour cela qu'il est important à bien programmer.

Voyons d'abord le problême du "bootage" de la disquette, qui permet de charger DOS avec une autre disquette que la disquette MASTER APPLE. D'autres problèmes peuvent surgir. Nous allons traiter de ce sujet dans cette partie.
Le lecteur peut initialiser une disquette comme il le désire. Par la suite nous laisserons au programme de bienvenue le nom: "HELLO". Ce nom est

largement répandu, il se trouve par exemple sur la disquette disquette système. HELLO délivre en premier des informations concernant la disquette insérée si elle est initialisée, son nom et son contenu. Exemple: on mémorise le programme suivant sur la disquette avec le nom: INITIAL.

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
5 PRINT"CE PROGRAMME SERT A L'INITIALISATION. IL FAUT DETRUIRE LA LIGNE 5
ET REMPLIR LA GRILLE DE LA LIGNE 20": END
10 TEXT: FOR I=1 TO 41: PRINT"*";: NEXT: GOSUB 999: HTAB 8 :PRINT"R.PRUST /
W.VOSS";: GOSUB 999
20 HTAB8: PRINT"DISQUETTE NR.  ";:GOSUB 999: HTAB 8:  PRINT"INITIALISEE LE
  .  .    ";
30 GOSUB 999:HTAB 40:FORI=1 TO 41: PRINT"*";: NEXT: PRINT: FOR I=1 TO 2500:
NEXT:HOME
40 VTAB 2:PRINT"CATALOG": END
999 HTAB 40: PRINT"**";: HTAB 40: PRINT"**";: RETURN
```

Lorsqu'on lance le programme, on voit apparaître le message suivant:

```
                APPLE TRUCS & ASTUCES

                R.PRUST / W.VOSS 1984

          CE PROGRAMME SERT A L'INITIALISATION
          IL FAUT DETRUIRE LA LIGNE 5
          ET REMPLIR LA GRILLE DE LA LIGNE 20
```

Après avoir effacé la ligne 5, on transforme de la manière la plus simple la ligne 20 avec le programme LINE EDITOR. Le programme est maintenant ainsi:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
10 TEXT: FOR I=1 TO 41: PRINT"*";: NEXT: GOSUB 999: HTAB 8 :PRINT"R.PRUST /
W.VOSS";: GOSUB 999
20 HTAB8: PRINT"DISQUETTE NR.  ";:GOSUB 999: HTAB 8:  PRINT"INITIALISEE  LE
  .  .    ";
30 GOSUB 999:HTAB 40:FORI=1 TO 41: PRINT"*";: NEXT: PRINT: FOR I=1 TO 2500:
NEXT:HOME
40 VTAB 2:PRINT"CATALOG": END
999 HTAB 40: PRINT"**";: HTAB 40: PRINT"**";: RETURN
```

Les lignes 10 à 30 produisent l'affichage suivant:

```
******************************
*                            *
*   R. PRUST / W. VOSS       *
*                            *
*   DISQUETTE NR.            *
*                            *
*  INITIALISEE LE   .  .     *
*                            *
******************************
```

Suit une boucle d'attente FOR...NEXT qui compte jusqu'à 2500 pour permettre de conserver l'écran un certain temps. Si on ne l'interrompt pas, le programme continue, d'abord avec HOME à la fin de la ligne 30. Puis la ligne 40 appelle la commande CATALOG de la disquette. Dans la chaîne, on entre CTRL D avant le C de CATALOG; il ne sera pas visible sur le listing de l'imprimante. On peut également entrer cette commande sous la forme:

```
PRINT CHR$ (4);"CATALOG".
```

La ligne 999 est un sous-programme, il y aura donc une petite étoile sur les bords d'impression gauche et droite de l'écran. On peut présenter l'écran autrement. Un programme initialisé disponible permet d'économiser du temps, surtout si il y a peu de modifications à effectuer sur la disquette initialisée. Si le programme INITIAL nécéssite de nombreuses informations, autres que la date et un numéro, il vaut mieux établir un programme qui demande toutes les données nécéssaires et qui écrive de son côté un programme c'est à dire un fichier texte INITIAL, qui pourra être chargée avec EXEC dans la mémoire. Comme ceci:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
10 REM PROGRAMME POUR L'ETABLISSEMENT
20 REM D'UN PROGRAMME D'INITIALISATION
30 INPUT"DATE (JJ.MM.AA)";A$: IF MID$(A$,3,1) "." OR MID$(A$,6,1) "."
THEN 30
40 INPUT"NR. DE DISQUETTE";N$
50 D$=CHR$(4)
60 PRINT D$;"OPEN INITIAL"
70 PRINT D$;"WRITE INITIAL"
80 PRINT "10 TEXT : FOR I=1 TO 41:?";CHR$(34);"*";CHR$(34);";:NEXT :GOSUB
999: HTAB 8: ?";CHR$(34);"R.PRUST";CHR$(34);";:GOSUB 999"
90 PRINT "20 HTAB 8:?";CHR$(34);"NR. DISQUETTE";N$;CHR$(34);";:GOSUB
999:HTAB 8:?";CHR$(34);" INITIALISEE LE";A$;CHR$(34);";"
100 PRINT "30 GOSUB 999:HTAB 40: FOR I=1 TO 41:?";CHR$(34);"*";
CHR$(34);";: NEXT: ?: FOR I=1 TO 2500: NEXT: HOME"
110 PRINT"42 VTAB 2:?";CHR$(34);"CATALOG"; CHR$(34);":END"
120 PRINT"999 HTAB 40:?";CHR$(34);"**";CHR$(34);";: HTAB 40:
?";CHR$(34);"**";CHR$(34);";:RETURN"
130 PRINT D$;"CLOSE INITIAL"
140 END
```

Ce programme MAKE INITIAL utilise un fichier de texte INITIAL de manière à

ce qu'après la commande EXEC INITIAL les lignes de 10 à 999 restent disponibles dans la mémoire centrale avec la différence que l'on place une date ou un numéro de disquette à la place du signe " ". Quelles sont les particularités du programme INITIAL ?

Description du programme:

Lignes	Contenu
10- 20	Commentaires
30	Entrée de la date sous la forme précisée dans A$; Petite routine de vérification
40	Entrée du numéro de disquette N$
50	Attribution de CTRL D à D$
60	Ouverture du fichier INITIAL.
70	Instruction à écrire dans le fichier. Toutes les commandes suivantes seront stockées sur la disquette.
80-120	Les lignes 10-999 du programme INITIAL seront inscrites sur la disquette comme un texte avec une instruction PRINT. Il faut que les guillemets soient placées avec leur code ASCII par CHR$(34) Avec l'introduction du programme MAKE INITIAL, les variables N$ et A$ sont remplacées par leur valeur initiale (lignes 30 et 40).
130	Le fichier est fermé
140	Fin du programme.

Le programme de bienvenue est maintenant disponible sur la disquette avec ses informations et ses commandes lorsque MAKE INITIAL se trouve sur la disquette:

```
RUN MAKE INITIAL
NEW
EXEC INITIAL.
```

On peut contrôler ceci avec LIST.

La commande NEW est nécessaire car avec EXEC l'ancien programme n'est pas effacé, à l'inverse de RUN. Seules les lignes de l'ancien programme dont les numéros concordent avec les numéros de ligne du nouveau programme, sont reportées, toutes les autres sont gardées. Sans NEW il se produit "un mélange de programmes" entre MAKE INITIAL et INITIAL, car le deuxieme programme est plus court que le premier. Pour que cela soit plus commode on peut continuer le programme MAKE INITIAL, la commande

```
RUN MAKE INITIAL
```

suffira à elle seule pour disposer de la partie désirée du programme de bienvenue initialisé sur une nouvelle disquette.

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
10 REM PROGRAMME POUR L'ETABLISSEMENT
20 REM D'UN PROGRAMME D'INITIALISATION
30 INPUT"DATE (JJ.MM.AA)";A$: IF MID$(A$,3,1) "." OR MID$(A$,6,1) "."
THEN 30
40 INPUT"NR. DE DISQUETTE";N$
50 D$=CHR$(4)
60 PRINT D$;"OPEN INITIAL"
70 PRINT D$;"WRITE INITIAL"
80 PRINT "10 TEXT : FOR I=1 TO 41:?";CHR$(34);"*";CHR$(34);";:NEXT :GOSUB
999: HTAB 8: ?";CHR$(34);"R.PRUST";CHR$(34);";:GOSUB 999"
90 PRINT "20 HTAB 8:?";CHR$(34);"NR. DISQUETTE";N$;CHR$(34);";:GOSUB
999:HTAB 8:?";CHR$(34);" INITIALISEE LE";A$;CHR$(34);";"
100 PRINT "30 GOSUB 999:HTAB 40: FOR I=1 TO 41:?";CHR$(34);"*";
CHR$(34);";: NEXT: ?: FOR I=1 TO 2500: NEXT: HOME"
110 PRINT"42 VTAB 2:?";CHR$(34);"CATALOG"; CHR$(34);":END"
120 PRINT"999 HTAB 40:?";CHR$(34);"**";CHR$(34);";: HTAB 40:
?";CHR$(34);"**";CHR$(34);";:RETURN"
```

```
130 PRINT D$;"CLOSE INITIAL"
140 PRINT D$;"EXEC INITIAL"
150 DEL 50,150
```

Les lignes de 10 à 120 sont identiques à la version précédente. Avec la commande ligne 130, le fichier de texte INITIAL (10-40,999) est chargé. Ensuite les lignes de l'ancien MAKE INITIAL qui n'ont pas été utilisées sont effacées ligne 140.
Après la commande DEL dans le programme, l'insertion du programme se termine et on évite de devoir rajouter ENT.

Après RUN MAKE INITIAL, on s'assure avec LIST que les lignes désirées se trouvent dans la mémoire centrale:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
10 TEXT : FOR I=1 TO 41:PRINT"*";:NEXT :GOSUB 999: HTAB 8: PRINT"R.PRUST /
W.VOSS";:GOSUB 999"
20 HTAB 8:PRINT "NR. DISQUETTE 17";:GOSUB 999:HTAB 8:PRINT" INITIALISEE LE
11.05.1984";
30 GOSUB 999:HTAB 40: FOR I=1 TO 41:PRINT"*";: NEXT: PRINT: FOR I=1 TO
2500: NEXT: HOME"
40 VTAB 2:PRINT"CATALOG":END"
999 HTAB 40: PRINT"**";:HTAB 40:PRINT"**";: RETURN
```

Avec ces messages dans le programme de bienvenue on peut ajouter l'EDITOR du chapitre 6 ou d'autres routines d'aide de la manière suivante:

Lorsque la mémoire centrale est vide on charge EDITOR avec:

```
        LOAD PROGRAM LINE EDITOR
```

Les lignes superflues sont effacées par

```
DEL 4,30
```

maintenant on peut charger le programme INITIAL avec EXEC INITIAL si la solution des fichiers texte à été choisie, ou avec APA et

```
&H
LOAD INITIAL
&M
```

Dans les 2 cas le programme se présente ainsi:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
10 TEXT : FOR I=1 TO 41:PRINT"*";:NEXT :GOSUB 999: HTAB 8: PRINT"R.PRUST
/W.VOSS";:GOSUB 999"
20 HTAB 8:PRINT "NR. DISQUETTE 17";:GOSUB 999:HTAB 8:PRINT" INITIALISEE LE
11.05.1984";
30 GOSUB 999:HTAB 40: FOR I=1 TO 41:PRINT"*";: NEXT: PRINT: FOR I=1 TO
2500: NEXT: HOME"
40 VTAB 2:PRINT"CATALOG":END"
999 HTAB 40: PRINT"**";:HTAB 40:PRINT"**";: RETURN
63999 CALL PEEK(175)+PEEK(176)*256-1822: RETURN: REM
```

```
* PROGRAM LINE EDITOR V2.0 *

      COPYRIGHT (C) 1980
        NEIL KONZEN
```

END est effacé ligne 40 et une nouvelle ligne 50 est insérée.

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
10 TEXT : FOR I=1 TO 41:PRINT"*";:NEXT :GOSUB 999: HTAB 8: PRINT"R.PRUST
/W.VOSS";:GOSUB 999"
20 HTAB 8:PRINT "NR. DISQUETTE 17";:GOSUB 999:HTAB 8:PRINT" INITIALISEE LE
11.05.1984";
30 GOSUB 999:HTAB 40: FOR I=1 TO 41:PRINT"*";: NEXT: PRINT: FOR I=1 TO
2500: NEXT: HOME"
40 VTAB 2:PRINT"CATALOG"
50 GOSUB 63999: PRINT"FP"
999 HTAB 40: PRINT"**";:HTAB 40:PRINT"**";: RETURN
63999 CALL PEEK(175)+PEEK(176)*256-1822: RETURN: REM
```

```
* PROGRAM LINE EDITOR V2.0 *

    COPYRIGHT (C) 1980
       NEIL KONZEN
```

Après le boot de la disquette, le programme de bienvenue HELLO rend EDITOR disponible.

La commande PRINT "FP" en ligne 50 termine le programme, elle remplace NEW et on revient au BASIC APPLESOFT.

Une fois la disquette initialisée avec HELLO ou un équivalent, on a la possibilité de travailler avec EDITOR tout de suite aprés la mise en route. La valeur de HIMEM est réduite. Aprés le boot du DOS et d'EDITOR , il ne

reste plus que 36 K disponibles. Pour construire d'autres routines d'aide dans HELLO ce n'est plus aussi simple. Pour pouvoir utiliser APA, il faut entrer la commande:

```
RUN LOAD APA
```

En outre les fichiers R BOOT, R LOAD et APA de LOAD APA qui doivent être à disposition sur la même disquette sont bootés.

Si l'on dispose de deux lecteurs, on peut procéder de la manière suivante: dès la mise en marche ou avec PR 6, le DOS TOOL KIT qui contient les 4 fichiers APA est inséré dans le lecteur 2.
Dans le lecteur 1 ,on place une disquette initialisée avec le HELLO suivant:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
10 TEXT : FOR I=1 TO 41:PRINT"*";:NEXT :GOSUB 999: HTAB 8: PRINT"R.PRUST
/W.VOSS";:GOSUB 999
20 HTAB 8:PRINT "NR. DISQUETTE 17";:GOSUB 999:HTAB 8:PRINT" INITIALISEE LE
11.05.1984";
30 GOSUB 999:HTAB 40: FOR I=1 TO 41:PRINT"*";: NEXT: PRINT: FOR I=1 TO
2500: NEXT: HOME
40 VTAB 2:PRINT"CATALOG"
50 GOSUB 63999: PRINT"RUN LOADAPA,D2"
999 HTAB 40: PRINT"**";:HTAB 40:PRINT"**";: RETURN
63999 CALL PEEK(175)+PEEK(176)*256-1822: RETURN: REM
```

```
* PROGRAM LINE EDITOR V2.0 *
```

COPYRIGHT (C) 1980
NEIL KONZEN

Les instructions PRINT contiennent dans les lignes 40 et 50, le signe correspondant à CTRL-D qui n'est pas visible sur le listing de l'imprimante. On commence par le bootage de l'affichage concernant le numéro de la disquette, le Catalog et le début d'EDITOR.

Le programme LOAD APA est lancé dans le deuxième lecteur, mais avec cette version de HELLO. Le programme de bienvenue se termine, LOAD APA se trouve dans la mémoire et le pointeur sur le deuxième lecteur. Pour éviter cela on peut modifier LOAD APA et la ligne 330. A la place de END, on envoie maintenant la commande CATALOG pour le Drive 1 et le programme peut être effacé par:

```
DEL 10, 360
```

Si une erreur s'est glissée pendant le chargement d'APA, la ligne 340 est sautée. La dernière partie du programme reste inchangée. Un nouveau départ est possible; END dans la ligne 360, est remplacé par DEL:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
100 REM PROGRAM TO LOAD APA
110 HOME
120 ONERR GOTO 340
130 ADRS=0
140 PRINT CHR$(4),"BLOAD RBOOT"
150 CALL 520: REM EXECUTE RBOOT
160 ADRS=USR(0),"APA"
170 REM BRING IN APA, ADRS=STARTING ADRESS
180 CALL ADRS: REM INITIALIZE APA
190 PRINT: PRINT
```

```
200 PRINT"& RENUMBER START , INC , FIRST , LAST "
210 PRINT"& HOLD"
220 PRINT"& MERGE"
230 PRINT"& LENGTH"
240 PRINT"& COMPRESS"
250 PRINT"& SHOW"
260 PRINT"& NOSHOW"
270 PRINT"& AUTO START , INC "
280 PRINT"& MANUAL"
290 PRINT"& XREF"
300 PRINT"& KEYS"
310 PRINT: PRINT
320 POKE 216,0: REM ON ERR OFF
330 PRINT"CATALOG,D1"
335 DEL 100,360
340 PRINT"UNABLE TO LOAD"
350 POKE 216,0: REM ON ERR OFF
360 END
```

Les lignes 190 et 310 contiennent toutes les commandes APA. On peut les détruire. Puis on peut mélanger ce LOAD APA raccourci avec le programme de bienvenue qui se présente maintenant comme ceci:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9):PRINT"R.PRUST / W.VOSS 1984":PRINT
10 TEXT : FOR I=1 TO 41:PRINT"*";:NEXT :GOSUB 999: HTAB 8: PRINT"R.PRUST
/W.VOSS";:GOSUB 999
20 HTAB 8:PRINT "NR. DISQUETTE 17";:GOSUB 999:HTAB 8:PRINT" INITIALISEE LE
11.05.1984";
30 GOSUB 999:HTAB 40: FOR I=1 TO 41:PRINT"*";: NEXT: PRINT: FOR I=1 TO
2500: NEXT: HOME
50 GOSUB 63999
100 REM PROGRAM TO LOAD APA
```

```
110 HOME
120 ONERR GOTO 220
130 ADRS=0
140 PRINT CHR$(4),"BLOAD RBOOT"
150 CALL 520: REM EXECUTE RBOOT
160 ADRS=USR(0),"APA"
170 REM BRING IN APA, ADRS=STARTING ADRESS
180 CALL ADRS: REM INITIALIZE APA
190 POKE 216,0: REM ON ERR OFF
200 PRINT"CATALOG,D1"
210 DEL 100,63999
220 PRINT"UNABLE TO LOAD"
230 POKE 216,0: REM ON ERR OFF
240 END
999 HTAB 40: PRINT"**";:HTAB 40:PRINT"**";: RETURN
63999 CALL PEEK(175)+PEEK(176)*256-1822: RETURN: REM
```

```
  * PROGRAM LINE EDITOR V2.0 *

       COPYRIGHT (C) 1980
          NEIL KONZEN
```

On lance EDITOR avec GOSUB 63999, ensuite on peut charger APA.

Si on ne dispose que d'un lecteur de disquettes, on peut remplacer les adresses du lecteur D1 par exemple, D2 dans les commandes de disquettes. Après avoir initialisé une disquette avec une des 2 dernières versions de HELLO, il faut copier les fichiers APA, R BOOT et R LOAD sur une nouvelle disquette, car LOADAPA les appelle (plus exactement RLOAD est appelé par RBOOT). Ceci peut se faire avec le File Developper du système de disquette par

C'est le style de programmation de l'utilisateur qui décidera, en dernier ressort, de la longueur du programme de bienvenue. Les fautes de frappe, qui se glissent toujours, seront corrigées par EDITOR. Pour le reste cela dépend des besoins de l'utilisateur, c'est pour cela que nous n'avons détaillés qu'un seul exemple pour APA qui explique le développement d'un programme d'initialisation.

Il faut penser que l'on perd de la mémoire avec le chargement immédiat d'APA que ce soit en mémoire centrale comme sur la disquette. Les routines doivent être compatibles les unes avec les autres. Il faut faire attention aux combinaisons tels que EDITOR et APA, lors du chargement.

7.5. Présentation de l'écran.

La présentation de l'écran n'est pas très complexe, mais elle demande du temps. Afin d'en gagner, voici quelques suggestions que nous avons relevées dans notre livre précédent.

7.5.1. Positionnement de l'écran

Nous présentons un résumé des commandes les plus importantes qui ont déjà été expliquées.

D'abord les commandes BASIC:

HTAB, VTAB, et la fonction TAB

comme complément de la commande PRINT. Ces instructions de positionnement ont déjà été utilisés dans les exemples établis jusqu'ici. On mettra simplement en évidence les avantages de HTAB et VTAB dans un petit exemple. Ces instructions ne sont pas disponibles sur tous les ordinateurs, l'APPLE permet une certaine aisance à ce niveau:

```
3 HOME : HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9): PRINT"R.PRUST / W.VOSS 1984":PRINT
10 HOME: PRINT TAB(10)"PROGRAMME DE DEMONSTRATION"
20 PRINT TAB(6)"POUR LA PRESENTATION DE L'ECRAN"
30 PRINT: PRINT: PRINT:PRINT: PRINT
40 PRINT TAB(25)"ENTREE:";:INPUT" ";A$
50 PRINT: PRINT: PRINT
60 PRINT"CECI DOIT SUFFIRE": PRINT: PRINT: PRINT
```

```
70 PRINT TAB(8)"AU REVOIR";TAB(25)"A BIENTOT"
80 END
```

Pour la position horizontale sur l'écran, on peut utiliser TAB; la verticale ne peut être déterminée que par rapport à la position du PRINT. TAB n'a d'effet que vers la droite comme dans la ligne 70. Si la valeur de la deuxième fonction TAB dans la ligne, est plus petite que la première, Il n'y a aucun déplacement vers la gauche et TAB est annulée.
HTAB et VTAB permettent le positionnement vertical et horizontal. Le numéro de ligne (entre 1 et 24) et le numéro de colonne (1 à 40) sont à spécifier: Ainsi quelques PRINT peuvent être remplacés:

```
3 HOME : HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9): PRINT"R.PRUST / W.VOSS 1984":PRINT
10 HOME: HTAB 10:PRINT"PROGRAMME DE DEMONSTRATION"
20 HTAB 6 :PRINT "POUR LA PRESENTATION DE L'ECRAN"
40 VTAB 8: HTAB 25: INPUT"ENTREE:";A$
60 VTAB 12:PRINT"CECI DOIT SUFFIRE"
70 VTAB 16: HTAB 8: PRINT"AU REVOIR";TAB(25)"A BIENTOT"
80 END
```

HTAB et VTAB peuvent être utilisés ensembles sur une même ligne. Les deux exemples suivants sont suffisamment clairs. On abaisse la vitesse d'édition pour mieux voir ce qui se passe:

```
3 HOME : HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9): PRINT"R.PRUST / W.VOSS 1984":PRINT
5 FOR I=1 TO 2000: NEXT
6 SPEED=100
10 HOME: HTAB 10:PRINT"PROGRAMME DE DEMONSTRATION"
20 HTAB 6 :PRINT "POUR LA PRESENTATION DE L'ECRAN"
40 VTAB 8: HTAB 25: INPUT"ENTREE:";A$
60 VTAB 12:PRINT"CECI DOIT SUFFIRE"
```

```
70 VTAB 22: PRINT"NOUS POUVONS ECRIRE QUELQUE CHOSE": PRINT"ET AUSSI
CONTINUER AU-DESSUS"
80 VTAB 12: CALL -868: HTAB 13: PRINT"VOILA QUI EST FAIT"
90 SPEED=255: END
```

CALL-868 permet d'effacer la partie droite d'une ligne après le curseur.

```
3 HOME : HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9): PRINT"R.PRUST / W.VOSS 1984":PRINT
5 FOR I=1 TO 2000: NEXT
6 SPEED=100
10 HOME: HTAB 10:PRINT"PROGRAMME DE DEMONSTRATION"
20 HTAB 6 :PRINT "POUR LA PRESENTATION DE L'ECRAN"
30 VTAB 15: PRINT"VOUS AVEZ LA POSSIBILITE AVEC VTAB D'ALLER VERS LE BAS DE
L'ECRAN": PRINT"ET REMONTER VERS LE HAUT POUR L'INPUT"
35 PRINT "MEME CHOSE QUE POUR HTAB": PRINT " C'EST CE QUE VOUS ALLEZ VOIR
BIENTOT"
40 VTAB 8:HTAB 25: PRINT"ENTREE:";
50 HTAB 9:PRINT"FAITES UNE ENTREE";
60 HTAB 34: INPUT" ";A$
70 VTAB 22: SPEED=255: END
```

7.5.2 Fonctions de chaînes

On ne peut pas parler de formation d'écran sans parler des fonctions de chaines.
Le contenu de l'écran n'est pas constitué de chiffres mais de textes commentés qui comprennent des chiffres.
Dans les exemples précédents, ces fonctions ont déjà été mises en place.
Dans l'exemple suivant on trouvera un petit jeu utilisant la rotation des caractères à l'intérieur d'une chaîne. Le premier caractère déplacé et mis en attente jusqu'à ce que la chaîne initiale soit repositionné.

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT: HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 HOME : INPUT"CHAINE ";X$
20 X$=X$+" ": X=LEN(X$)
30 HOME : PRINT X$
40 FORI=1 TO X-1
50 Y$=LEFT$(X$,1)
60 X$=RIGHT$(X$,X-1)+Y$
70 PRINT X$
80 NEXT
90 END
```

L'impression à l'écran de la chaîne "ANNIVERSAIRE " donne ceci:

```
ANNIVERSAIRE
NNIVERSAIRE A
NIVERSAIRE AN
IVERSAIRE ANN
VERSAIRE ANNI
ERSAIRE ANNIV
SAIRE ANNIVER
AIRE ANNIVERS
```

```
RE ANNIVERSAI
E ANNIVERSAIR
 ANNIVERSAIRE
```

Voici la version améliorée de ce programme:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT: HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 HOME : INPUT"CHAINE ";X$
20 X$=X$+" ": X=LEN(X$)
22 IF X 22 THEN PRINT: INVERSE: PRINT"ATTENTION:": NORMAL: PRINT: PRINT"LA
CHAINE DOIT ETRE COMPOSEE DE MOINS DE 22 CARACTERES": CALL -198: FOR I=1 TO
2000: NEXT: GOTO10
25 A=21-INT(X/2)
26 B=12-INT(X/2)
30 HOME: VTAB B: HTAB A: PRINT X$
40 FORI=1 TO X-1
50 Y$=LEFT$(X$,1)
60 X$=RIGHT$(X$,X-1)+Y$
70 HTAB A:PRINT X$
80 NEXT
90 END
```

L'affichage occupe autant de lignes que de caractères dans X$. Pour des chaînes de plus de 22 caractères l'affichage serait incomplet, c'est pourquoi la ligne 22 est ajoutée, elle envoie un message d'erreur le cas échéant.

La routine CALL-198 émet un "BIP". Le texte doit être centré; les lignes 25 et 26, établissent les variables A (pour la position horizontale) et B (pour la position verticale).
La longueur de la variable X$ donne le nombre de lignes et de colonnes écran libres. Dans les lignes 30 et 70 on ajoute les commandes HTAB et VTAB

correspondantes.

Dans ce programme il sera expliqué comment les fonctions de chaînes sont utilisées pour rechercher des chaînes et les analyser. Ce type de technique est nécessaire non seulement pour des jeux mais pour des traitements de textes et des gestions de fichiers. Il faut préciser que le positionnement du texte peut être calculé par l'ordinateur grâce à certaines instructions.

7.5.3 Sorties d'écran formatées

Pour l'utilisateur du système d'exploitation CP/M le formatage des sorties écran n'est pas un problème grâce à la commande PRINT USING. Malheureusement cette commande n'est pas valide sur l'APPLE II. C'est pourquoi nous allons donner des solutions à ce problème.

Une sortie formatée signifie:
Création d'une zone de texte et d'une zone dans laquelle les nombres sont placés les uns sous les autres, les points décimaux étant alignés et les parties décimales éventuellement manquantes remplacées par des zéros.
Les chaînes comme les nombres doivent habituellement être placés à gauche de l'écran. Ceci sera expliqué dans le programme suivant. Celui-ci écrit 20 nombres dans une matrice de 5 lignes et 4 colonnes, la position de départ étant précisée dans HTAB.

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT: HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 HOME: FOR I=0 TO 4:FOR J=0 TO 3: READ X(I,J): NEXT J,I
20 FOR I=0 TO 4
30 FOR J=0 TO 3: HTAB (J*9+1): PRINT X(I,J);: NEXT
40 PRINT : NEXT I
999 END
1000 DATA 12.2345,3,7.8,1789.4
1010 DATA 4,121.789,-23.5,4.121
1020 DATA 1,2,3,4
1030 DATA 65.4386,34218,3.764,9
1040 DATA 0.8,198,34.274,456.1
```

L'impression est ainsi:

```
    12.2345  3         7.8       1789.4
```

```
4         121.789   -23.5     4.121
1         2         3         4
65.4386   34218     3.764     9
.8        198       34.274    456.1
```

Comment obtenir une impression telle que celle-ci ?

```
 12.23      3.00      7.80    1789.40
  4.00    121.78    - 23.50      4.12
  1.00      2.00      3.00      4.00
 65.43   1234218      3.76      9.00
  0.80    198.00     34.27     456.10
```

Les nombres sont donnés 2 chiffres après la virgule et placés à droite dans les zones établies auparavant.
Pour les nombres compris entre -1 et +1 on ajoute un zéro. Si un nombre apparaît entre deux ,c'est qu'il était trop long pour la zone donnée.
Le programme suivant est plus long que le précédent:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT : PRINT: HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 HOME : DIM X(4,3),F$(3)
20 DEF FN R(X)=INT(X*100+.5)/100
30 FOR I=0 TO 4: FOR J=0 TO 3: READ X(I,J): NEXT J,I
40 FOR J=0 TO 3: READ F$(J): NEXT
50 FOR I=0 TO 4
60 FOR J= 0 TO 3: X=FN R(X(I,J)): X$=STR$(X)
70 IF LEFT$(X$,1)  "-" THEN X$=" "+X$
80  L=LEN(X$):  IF  MID$(X$,2,1)="."  THEN  X$=  LEFT$  (X$,1)+"0"+  RIGHT$
(X$,L-1): L=L+1
90 IF L=2 THEN L=L+1
100 IF MID$(X$,L-2,1)="." THEN 130
110 IF MID$(X$,L-1,1)  "." THEN X$=X$+".0"
```

```
120 X$=X$+"0": L=LEN(X$)
130 F=LEN(F$(J)): IF L F THEN X$=" "+MID$(X$,2,F-2)+" ": GOTO160
140 IF L=F THEN 160
150 X$=LEFT$(X$,1)+LEFT$(F$(J),F-L)+RIGHT$(X$,L-1)
160 HTAB (J*10+1): PRINT X$;: NEXT J
170 PRINT : NEXT I
180 END
1000 DATA 12.2345,3,7.8,1789.4
1010 DATA 4,121.789,-23.5,4.121
1020 DATA 1,2,3,4
1030 DATA 65.4386,34218,3.764,9
1040 DATA 0.8,198,34.274,456.1
1100 DATA "       ","       ","       ","       "
```

Description du programme

Lignes	Description
10	Les tableaux X et F$ sont dimensionnés
20	Fonction d'arrondissement R
30	Lecture des 5*4 nombres dans le tableau X
40	Etablissement des 4 zones avec F$
50	Boucle I pour lignes
60	Boucle J pour zones Un nombre est arrondi deux chiffres après le point décimal puis placé dans X$
70	Si 0 la chaîne X$ prend un espace au début
80	Attribution de la variable L à la longueur de la chaîne X$ Teste si le nombre est compris entre -1 et +1 et 0

	Ce nombre sera formaté avec le point décimal à la deuxième position p.ex. "-.3" ou " .3" On ajoute enfin un 0 au début de manière à avoir une impression du type "0.3". L doit être 1
90	Pour un nombre entier L=2 Dans la ligne 100 pour éviter une erreur dans MID$ on (L-2=0) on incrémente L
100-120	En cas de besoin on ajoute des zéros après le point décimal 100 Si le nombre arrondi a deux chiffres après la virgule le 3ème signe de X$ est un point, on saute les deux chiffres suivants. 110 Si le nombre est un entier (. en dernière position ou décimal à un chiffre (. en avant dernière position) on ajoute ".0" 120 On complète avec un 0 les nombres entiers et les nombres à un chiffre après la virgule On cherche la longueur de la chaîne
130-150	On place X$ dans F$(J) 130 On vérifie si F L sinon deux sont mis à la place des premiers et derniers caractères de X$ et on se branche sur 160. 140 Si L=F on se branche sur 160 et aucune modification n'est nécessaire. 150 X$ est placé à droite dans F$(J) Si X 0 on saute la première place de la zone Le reste est complété par les espaces de F$(J)
160	Le début de la zone est placé avec HTAB et l'impression commence.
170	Après un interlignage (à cause du point-virgule

	dans le PRINT de la ligne 160), fin de boucle I
180	Fin du programme
1000-1100	DATA pour commande READ des lignes 30 et 40

Ces sorties formatées demandent un peu de temps. Mais il peut être nécessaire d'avoir un programme tel que celui-ci comme sous-programme en le fusionnant à un programme pricipal par la méthode EXC ou APA (il faudra fixer une valeur finale pour I et J).

7.5.4 Les masques d'écran

Ce qu'on entend par masque d'écran c'est le partage en zones de l'écran pour la saisie des données (nombres et textes) qui doivent être traitées dans le programme ou stockées dans un fichier. Pour cela il est important de préciser à l'utilisateur ce qui doit être entré et sous quelle forme. Voici une mise en place de masques d'écran:

```
3 HOME : HTAB(9): PRINT "APPLE TRUCS & ASTUCES"
4 PRINT: PRINT : HTAB(9): PRINT"R.PRUST / W.VOSS 1984": PRINT
10 HOME : HTAB 6: PRINT"RENSEIGNEMENTS PERSONNELS" : PRINT "FOR Q=1 TO 40:
PRINT"-";: NEXT : PRINT: POKE 34,4: HOME
20 PRINT "NOM:.............";: HTAB 22: PRINT"PRENOM:............."
30 VTAB 8: FOR Q=1 TO 34: PRINT ".";: NEXT: HTAB 36: PRINT"...."
40 HTAB 14: PRINT"RUE";:HTAB 37:PRINT"NO."
50 VTAB 12: PRINT".....";: HTAB 9: FOR Q=1 TO 20: PRINT".";: NEXT :PRINT
60 PRINT"CODE";: HTAB 15: PRINT"VILLE"
70 VTAB 16: PRINT".....";: HTAB 7 : PRINT".....";: HTAB 24:
PRINT"............."
75 PRINT"AGE";: HTAB 7 : PRINT"TRANCHE D'IMPOT";: HTAB 24: PRINT"NOMBRE
D'ENFANTS"
80 PRINT: HTAB 15: PRINT"NOMBRE D'ENFANTS:...."
90 VTAB 19: PRINT"FAITES VOS ENTREES": PRINT"LA OU LE CURSEUR CLIGNOTE ":
PRINT"JUSQU'A LA LIMITE": PRINT"PUIS APPUYEZ SUR RETURN "
100 PV=5: PH=5: GOSUB 1000: N$=X$: PH=29: GOSUB 1000: VN$=X$
110 PV=8: PH=1: GOSUB 1000: S$=X$: PH=36: GOSUB 1000: NR$=S$
120 PV=12: PH=1: GOSUB 1000: PL$=X$: PH=9: GOSUB 1000: W$=X$
130 PV=16: PH=1: GOSUB 1000: A$=X$: PH=8 : GOSUB 1000: SK$=X$: PH=24: GOSUB
1000: KZ$=X$
140 VTAB 19: CALL -958: HTAB 6: INVERSE: PRINT "APPUYEZ SUR N'IMPORTE
QUELLE TOUCHE": NORMAL : POKE -16368,0: WAIT -16384,128: POKE -16368,0:
HOME
150 END
```

```
1000 X$="": VTAB PV
1010 HTAB PH: GET E$: PRINT E$;:IF E$ CHR$(13) THEN PH=PH+1:X$=X$+E$:
GOTO1010
```

Description du programme:

Lignes	Description
10	Impression du titre et limite des 4 lignes en haut de l'écran avec POKE 34,4
20-80	Intitulés des champs d'entrée
90	Indication sur les entrées
100-130	Entrée des données Après le positionnement horizontal et vertical on se branche sur le s.p. 1000 puis X$ est transmis au programme principal
140	Interruption du programme avec WAIT
150	Dans l'exemple le programme se termine ici. Il pourrait y avoir d'autres instructions pour le traitement des données.
1000	Sous-programme pour l'entrée. On vide la chaîne X$ et on détermine la position verticale. PV doit être transmis par le programme principal PH l'est dans la ligne suivante
1010	Un caractère est attendu puis affiché jusqu'à ce qu'on appuie sur RETURN. Les caractères sont stockés dans X$, la position horizontale incrémentée.
1020	Retour au programme principal. E$=CHR$ (13) dès que l'on appuie sur la touche RETURN.

Il manque dans ce programme des possibilités de corrections d'erreurs. L'utilisateur peut entrer des données d'une certaine longueur, mais il n'y

a pas de commandes dans le programme qui permettraient d'éviter certaines erreurs. La plausibilité des entrées n'est pas vérifiée. On utilise des chaînes pour toutes les entrées, les suites alphabétiques pour l'âge ou le numéro de la maison sont valables.

Pour une programmation plus fine, il faut utiliser un générateur de masques. Le programme nécessite plus de 3 pages de listings. Il met en place un fichier EXEC avec le nom MASQUE qui contient les lignes de programme. Les numéros peuvent être choisis de manière quelconque, ils sont insérés dans le programme principal avec la commande EXEC et peuvent être utilisés dans plusieurs pages. Un de ses grands avantages est que l'on peut voir le programme en cours de déroulement sur l'écran, sans la commande PRINT. Il faut simplement indiquer le champ d'entrée avec un signe spécial. L'écran est placé en 1048/2047.

Le générateur de masque a tout de même un léger inconvénient, les entrées ne sont pas vérifiées pour les fautes de syntaxe. Aussi on n'utilisera pas le programm MASQUE pour les fautes de syntaxe.

Nous allons reprendre le programme du générateur de masque de FRANCK BRALL. Les modifications apportées ne concernent que les fautes de frappe et non pas le déroulement du programme ni de la syntaxe. Nous avons pris le listing dans la revue "COMPUTRONIC" avril 84. Nous ne décrirons pas le programme car il nous est étranger.
Nous allons clore ce chapitre avec cet exemple important:

```
80 ONERR GOTO 1360
90 HOME
100 PRINT"GENERATEUR DE MASQUE D'ECRAN"
110 PRINT : PRINT"    (C) PAR FRANK BRALL"
120 PRINT: PRINT
130 PRINT"CE PROGRAMME GENERE UNE INTERRUPTION": PRINT"CE MASQUE EST
PRODUIT EN  BASIC"
170 PRINT: PRINT"EN APPUYANT SUR UNE TOUCHE UN EXEMPLE EST MIS EN PLACE PAR
L'EDITEUR"
180 GET E$: IF E$="" THEN 180
190 HOME : PRINT"NO DE CARTE.:      DATE:   ":
PRINT"----------------------------------------": PRINT :
PRINT"NOM:             PRENOM:            ": PRINT"DERNIERE
ADRESSE:                          "
200 PRINT : PRINT: PRINT"ADRESSE ACTUELLE: CP                ": PRINT:
PRINT:PRINT"REMARQUES:                  ": PRINT:
PRINT"                                      "
210 GET E$: IF E$="" THEN 210
220 HOME
310 DIM X(100),Y(100),V$(100),L(100)
320 FOR I=768 TO 880 : READ E: POKEI,E: NEXT I
330 DATA 164,36,169,32,145,40,173,16,192,173,0
,192,16,251,72,164,36,177,40,73,128,145, 40,104,201,133,208,1
,96,201,131,208,6 ,32,88,252,76,0,3,201,141,240,10,201,136,240,
12,166,36,224,39,208,6
340 DATA 166,37,224,23,240,43,201,149,208,16,166,
36,224,39,208,5,169,141,76, 79,3,230,36,76,102, 3,201,129,208,
6,32,26,252,76,102,3,201,154,208,6,32,102,252,76,102,
3,32,237,253,264,36,177,73,128,145,40,76,6,3
350 DATA 1024,1152,1280,1408, 1536,1664,1792,1920,1064,
1192,1320,1448,1576,1704,1832,1960, 1104,1232,1360,1488,1616,1744,1872,
2000
360 INPUT"PREMIER NO DE LIGNE? ";ZN : HOME
```

```
370 PRINT"EDITEUR DE MASQUE":
PRINT"----------------------------------------"
380 PRINT: PRINT: PRINT"CTRL-A =CURSEUR LIGNE 1"
390 PRINT: PRINT"CTRL-Z =CURSEUR LIGNE SUIVANTE"
400 PRINT: PRINT"  -    =CURSEUR EMPLACEMENT SUIVANT"
410 PRINT: PRINT"   -   =CURSEUR EMPLACEMENT PRECEDENT"
420 PRINT: PRINT"RETURN =DEBUT DE LIGNE"
430 PRINT: PRINT"CTRL-C =EFFACER L'ECRAN"
440 PRINT: PRINT"CTRL-E =MASQUE"
450 PRINT: PRINT"       =SIGNE DE RECONNAISSANCE"
460 PRINT: PRINT"        LA FENETRE EST DESSINEE"
470 PRINT: CALL 768: GET E$
480 POKE 36,0: POKE 37,0: PRINT
490 D$=CHR$(4): PRINT D$;"DELETE MASQUE"
500 CALL 1002
510 PRINT D$;"OPEN MASQUE"
520 PRINT D$;"WRITE MASQUE"
530 PRINT ZN;"REM **** MASQUE D'ECRAN ****": ZN=ZN+1
540 PRINT ZN;"GOTO";ZN+21: ZN=ZN+1: NR=ZN
550 PRINT ZN;"VTAB(TY):HTAB(TX): INVERSE : FOR ZE=1 TO LE: ?CHR$(32);:NEXT"
:ZN=Z+1
560 PRINT ZN;"VTAB(YTY):HTAB(TX): GE$="; CHR$(34);CHR$(  );":ZE=0":ZN=ZN+1
570 PRINT ZN;"GET EI$: IF EI$  CHR$(13) AND EI$  CHR$(21)THEN";ZN+3:
ZN=ZN+1
580 PRINT ZN;"IF EI$=CHR$(21) THEN ";ZN-1:ZN=ZN+1
590 PRINT ZN;"GE$=LEFT$(ME$,LE) : GOTO";ZN+13:ZN=ZN+1
600 PRINT ZN;"FOR ZE=1 TO LE: ?CHR$(32);:NEXT ZE: VTAB(TY): HTAB(TX): ZE=0:
GOTO";ZN+2: ZN=ZN+1
610 PRINT ZN;"GET EI$": ZN=ZN+1
620 PRINT ZN;"IF EI$=CHR$(27) THEN GE$=LE$: GOTO";ZN=ZN+1
630 PRINT ZN;"IF EI$=CHR$(13) THEN ";ZN+8: ZN=ZN+1
640 PRINT ZN;"IF EI$=CHR$(21) THEN ";ZN-3: ZN=ZN+1
650 PRINT ZN;"IF EI$=CHR$(8) AND ZE=0 THEN ";ZN-4: ZN=ZN+1
```

```
660 PRINT ZN;"IF EI$=CHR$(8) THEN ZE=ZE-1: GOTO ";ZN+3: ZN=ZN+1
670 PRINT ZN;"IF ZE=LE-1 THEN GE$=GE$+EI$: ZE=ZE+1: GOTO";ZN: ZN=ZN+1
680 PRINT ZN;"ZE=ZE+1": ZN=ZN+1: PRINT ZN;"? EI$;:GE$=GE$+EI$": ZN=ZN+1
690 PRINT ZN;"GOTO";ZN-9: ZN=ZN+1
700 PRINT ZN;"IF GE$  ";CHR$(34);CHR$(34)" THEN GE$=LEFT$(GE$,ZE)":ZN=ZN+1
710 PRINT ZN;"NORMAL:VTAB(TY): HTAB(TX): FOR ZE=1 TO LE: ?CHR$(32);:NEXT :
ZN=ZN+1
720 PRINT ZN;"VTAB(TY):HTAB(TX):?GE$;:RETURN":ZN=ZN+1
730 PRINT ZN;"HOME": ZN=ZN+1
740 FOR U=1 TO 24
750 READ A1: ST$=""
760 FOR I=A1 TO A1+39 : C=PEEK(I)-28
770 IF C=64 AND F=0 THEN F=1: X(Z)=I+1-A1: Y(Z)=Z+1
780 IF C  64 AND F=1 THEN F=0: L(Z)=I+1-A1: Y(Z)=Z+1
790 IF C=64 OR C=34 THEN C=32
800 ST$=ST$+CHR$(C)
810 NEXT I
830 IF F=1 THEN F=0: L(Z)=I+1-A1-X(Z): Z=Z+1
840 I=0 : T=40
850 IF MID$(ST$,T,1)  ""THEN 870
860 I=I+1: T=T-1: IF T 0 THEN 850
870 IF U=24 AND I 2 THEN I=2
880 IF U=24 AND I=40 THEN PRINT ZN;"PRINT";CHR$(34);CHR$(34);";":GOTO 930
890 IF I=40 THEN PRINT ZN;"PRINT": GOTO 930
900 IF I=0 THEN PRINT ZN;"PRINT";CHR$(34);ST$;CHR$(34);";":GOTO 930
910 IF U=24 THEN PRINT ZN;"PRINT";CHR$(34);:PRINT LEFT$(ST$,41-I);: PRINT
CHR$(34);";":GOTO 930
920 PRINT ZN;"PRINT";CHR$(34);:PRINT LEFT$(ST$,40-I);:PRINT CHR$(34)
930 ZN=ZN+1 : NEXT U
940 PRINT D$;"CLOSE MASQUE"
950 IF Z=0 THEN HOME: PRINT"ERREUR!": PRINT: PRINT"IL MANQUE LE SIGNE DE
RECONNAISSANCE POUR LA FENETRE!": PRINT: PRINT: PRINT"EXEMPLE:
DATE:       ":PRINT:PRINT"          NOM:              NR.  ":END
```

```
960 VTAB24: HTAB 1:PRINT"                                        ";:U=24
970 FOR Q=0 TO Z-1
980 IF U=24 AND Y(Q)=24 THEN U=22: VTAB24: HTAB1:PRINT LEFT$(ST$,39);
990 FLASH
1000 VTAB(Y(Q)): HTAB(X(Q)): PRINT" ";
1010 NORMAL
1020 VTABU:HTAB1: PRINT "NOM DE LA VARIABLE";Q+1;"?     ";
1030 G$="": HTAB18
1040 GET E$
1050 IF E$=CHR$(8) THEN 1020
1060 IF E$  CHR$(13) THEN PRINT E$;: G$=G$+E$: GOTO 1040
1070 V$(Q)=G$: IF G$=CHR$(13) THEN 1020
1080 HTAB(X(Q)): VTAB(Y(Q)): PRINT" ";
1090 NEXT Q
1100 HOME : PRINT D$;"MON 0"
1110 PRINT D$;"APPEND MASQUE"
1120 PRINT D$;"WRITE MASQUE"
1130 FOR Q=0 TO Z-1
1140 F=0
1150 FOR I=1 TO LEN(V$(Q)): MID$(V$(Q),I,1)="$" THEN F=1
1160 NEXT I
1170 IF F=1 THEN PRINT ZN;"VTAB(";Y(Q);"):HTAB(";X(Q);"):?
LEFT$(";V$(Q);",";L(Q);");"
1180 IF F=0 THEN PRINT
ZN;"VTAB(";Y(Q);"):HTAB(";X(Q);"):ME$=STR$(";V$(Q);"):?
RIGHT$(ME$,";L(Q);");"
1190 ZN=ZN+1: NEXT Q: F=0
1200 FOR Q=0 TO Z-1
1210 PRINT ZN;"TX=";X(Q);":TY=";Y(Q);":LE=";L(Q):ZN=ZN+1
1220 FOR I=1 TO LEN(V$(Q)): IF MID$(V$(Q),I,1)="$" THEN F=1
1230 NEXT I
1240 IF F=1 THEN PRINT ZN;"ME$";V$(Q);":GOSUB ";NR:ZN=ZN+1
1250 IF F=1 AND Q 0 THEN PRINT ZN;"IF EI$=CHR$(27) THEN ";ZN-6:ZN=ZN+1
```

```
1260 IF F=1 THEN PRINT ZN;V$(Q);"=GE$": ZN=ZN+1
1270 IF F=0 THEN PRINT ZN;"ME$=RIGHT$(STR$(";V$(Q);"),";L(Q);"):GOSUB
;"NR;":IF GE$=";CHR$(34);CHR$(34);"THEN ";ZN:ZN=ZN+1
1280 IF F=0 AND Q 0 THEN PRINT ZN;"IF EI$=CHR$(27) THEN ";ZN-6: ZN=ZN+1
1290 IF F=0 THEN PRINT ZN;V$(Q);"=VAL(GE$)":ZN=ZN+1
300 F=0: NEXT Q
1310 PRINT ZN;"REM * FIN *"
1320 PRINT D$;"CLOSE MASQUE"
1330 HOME : PRINT" LE MASQUE EST MIS EN PLACE":PRINT: PRINT: PRINT"AVEC LA
COMMANDE";
1340 INVERSE : PRINT"EXEC MASQUE": NORMAL: PRINT" ET EST MEMORISE"
1350 END
1360 REM *** CONTROLE D'ERREUR ***
1370 QE=PEEK(222)
1380 IF QE=6 THEN 500
1390 PRINT CHR$(7): PRINT D$;"CLOSE MASQUE"
1400 PRINT: INVERSE : PRINT"CODE ERREUR ";QE: NORMAL:END
```

7.6. Classement.

Dans le chapitre 1, il y a un exemple de programme de classement. Ce programme est appelé BUBBLE, chaque élément du tableau est comparé avec le suivant et permuté lorsque l'ordre n'est pas juste. Ce programme nécessite deux boucle emboitées. C'est un moyen sûr de programmer, mais il est long car pour un tableau de dimension N, il faut effectuer N(N-1)/2 comparaisons. Pour N=100 il y a déjà 4950 comparaisons. Nous ne pouvons pas dire combien de temps cela dure car cette opération dépend du nombre de permutations. La plupart des routines de classement utilise de tels procédés et après avoir testé différents programmes nous pensons que c'est le meilleur algorythme de classement. Certaines routines ne sont développées que pour des ordinateurs importants, et pour les petits ordinateurs il vaut mieux utiliser la méthode de comparaison directe. Le temps d'éxécution étant défini par le système d'exploitation, le même programme ne peut pas être valable pour tous les ordinateurs.

Les routines de classements sont divisées en 2 groupes:

Tri-X et Tri-X/Y.

X-Tri: Les programmes classent un tableau (X) en ordre croissant ou décroissant.

Tri-X/Y: Trie un tableau (X) et stocke les valeurs correspondantes dans un deuxième tableau Y.

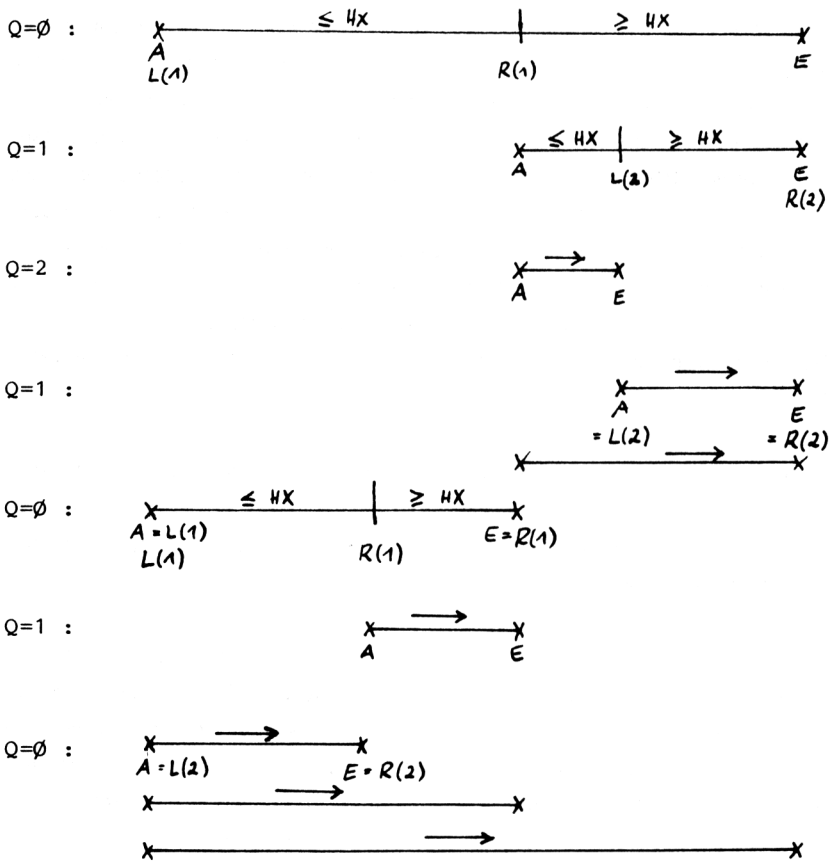
L'APPLE dispose pour le premier cas de SUPER QUICK SORT de KOLBE et pour le deuxième, un Tri X/Y de VELLEMAN et HOAGLIN.

Les programmes de tri sont construits d'une manière compliquée, mais basée sur une idée simple: par exemple le classement de documents d'après l'alphabet ou par sujets. Cela permet de gagner un temps appréciable et d'établir un classement fini: ainsi tous les documents sont classés d'après leurs initiales, ensuite ceux qui ont les mêmes lettres. La routine procède

de la mème façon que l'algorythme de VELLEMAN et HOAGLIN. Le tableau entier est classé pour que les valeurs de la première partie soient toutes inférieures ou égales à une valeur de comparaison (HX) et les valeurs de la deuxième partie supérieures ou égales à celle-ci. La valeur de comparaison est la valeur X(N/Z) du tableau. Si N/Z n'est pas entier l'ordinateur prend automatiquement comme index la valeur immédiatement inférieure. Le morceau le plus long est ensuite repéré à l'aide des variables L(Q) et R(Q) qui correspondent au début et à l'index de fin de cette partie et sont mémorisés. Q donne le niveau de décomposition. Le segment le plus court est désigné par A et ses éléments sont comparés à une nouvelle valeur moyenne HX.

Les éléments plus petits ou égaux à HX et plus grands ou égaux à HX sont regroupés. Q est incrémenté, le segment le plus important est mémorisé, la comparaison continue avec le plus petit. Le segment ne doit pas contenir plus de quinze éléments.

Ceci dure aussi longtemps que le plus petit segment contient au plus 15 éléments. Ils sont classés (lignes 30 et 40). Avec Q=Q-1 on retourne un niveau en arrière. Si le segment dépasse 15 éléments, on recommence le processus, ainsi le classement final peut avoir lieu en lignes 30 et 40. Les parties classées d'un niveau sont jointes les unes aux autres, car elles ont été séparées par les mèmes valeurs de comparaison.
Représentons ceci schématiquement:



Une flèche signifie que la zone correspondante du tableau est classée.

Le pré-classement se fait dans les lignes 50 à 80, les lignes 90/140 déterminent quelle partie sera mémorisée et quelles sont les nouvelles limites.

En ligne 150 on vérifie que Q=0 ce qui veut dire que le tableau est classé. Si non on procède à un autre niveau de décomposition, et Q est réduit:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9): PRINT"R.PRUST / W. VOSS 1984": PRINT
10 A=0: E=N: H=14: Q=0
20 IF E-A H THEN 50
30 FOR I=A TO E-1: Y=X(I): K=I: FOR J=I+1 TO E: IF X(J) Y THEN Y=X(J): K=J
40 NEXT: X(K)=X(I): X(I)=Y: NEXT: GOTO 150
50 I=A: J=E: HX=X((I+J)/2)
60 FOR I=I TO E: IF X(I) HX THEN NEXT
70 FOR J=J TO A STEP-1: IF X(J) HX THEN NEXT
80 IF I J THEN Y=X(I): X(I)=X(J) : X(J)=Y : I=I+1: J=J-1: IF I J THEN 60
90 IF J-A E-I THEN 130
100 IF I E THEN Q=Q+1: L(Q)=I: R(Q)=E
110 E=J: IF A E THEN 20
120 GOTO 150
130 IF J A THEN Q=Q+1: L(Q)=A: R(Q)=J
140 A=I: IF A E THEN 20
150 IF Q 0 THEN A=L(Q): E=R(Q): Q=Q-1: GOTO 20
160 END
```

Le TRI X/Y procède d'une manière plus lente mais moins compliquée:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9): PRINT"R.PRUST / W. VOSS 1984": PRINT
10 L=N-1
20 L=INT((L-2)/3)+1: FOR I=1 TO N-L: J=I+L: HX=X(J): H(Y)=Y(J): IF X(I) =HX
THEN 70
```

```
30 Q=I
40 X(J)=X(Q): Y(J)=Y(Q): J=Q: IF Q =L THEN 60
50 Q=Q-L: IF X(Q) HX THEN 40
60 X(J)=HX: Y(J)=HY
70 NEXT I: IF L 1 THEN 20
80 END
```

Le tableau initial X est divisé en trois parties (la valeur du milieu peut être éventuellement un plus grande quand N n'est pas divisible par 3).

Maintenant on compare le premier élément de la première partie avec le premier élément de la deuxième partie, il y a éventuellement permutation; puis on compare le deuxième élément de la première partie avec le deuxième élément de la deuxième partie etc....L'intervalle entre les éléments comparés est toujours égal à L.
Ceci est valable après le premier passage pour chaque I de la première zone:
X(I) = X(I+L).

Maintenant les éléments de la deuxième partie sont comparés avec les éléments correspondants de la troisième partie, donc avec l'intervalle L. Le plus important des deux appartient au dernier tiers; le plus petit des deux sera comparé avec ceux du premier tiers et éventuellement permuté. Ceci est valable pour tous les I de la première partie:
Pour tout I =L: X(I) = X(I+L) = X(I+2L)

L est rapproché le plus possible d'un nombre divisible par 3. Par exemple:

Pour N=200, L est placé sur 199 (Ligne 10) puis on le calcule: L=INT(199-2)/3+1=INT(197/3)+1=INT(65.666)+1; 65+1=66.

Lorsqu'on a 198/3, 198 est divisible par 3 le plus proche de 199. Pour chaque nombre entier il y a un nombre voisin divisible par 3 ou multiple de

Lors du premier passage de la boucle dans les lignes 20 à 70, L pour N =200 a la valeur 66. Cette valeur est réduite d'un tiers dans la ligne 20. 66 est divisible par 3, L devient 22. Le tableau est décomposé en 3 parties et ainsi de suite. Chaque élément d'une partie quelconque est comparée avec un élément correspondant. Après le premier passage de la boucle on a 66 rangées ordonnées d'une longueur de 3 qui ont été permutés les uns derrière les autres.
Le plus proche du nombre 22 divisible par 3 est 21, L est réduit à 7 il y a 28 segments. De 7 L passe à 2, chaque élément est comparé avec le suivant. Après ce passage il ne reste plus que 2 séries à ordonner placées l'une derrière l'autre.
Si l'on réduit L à 1 la boucle se déroule encore 1 fois, les éléments voisins seront comparés et on obtiendra 1 rangée ordonnée.

Dans les explications précédentes nous n'avons parlé que d'un tableau, car le tableau Y n'est que chargé et ses éléments sont introduits lorsque les valeurs du tableau X sont déplacés dans les lignes 40 et 60.

Nous allons clore cette partie avec un exemple moins théorique. Les routines sont développées pour des nombres mais elles peuvent également comporter des chaînes.

On désire établir un index avec des numéros de page de manière à pouvoir le consulter. Les mots doivent être classés par ordre alphabétique et les numéros doivent suivre. Ensuite tous les numéros de page concernant le même mot sont classés par ordre de grandeur.

Les deux routines sont mises en oeuvre. Dans le TRI X/Y, X est remplacé par X$, dans le TRI X, X est remplacé par Y: ces transformations se trouvent dans le programme REPLACE de l'APPLE DOC . Ils sont construits comme des sous-routines. Il est vraisemblable que pendant le passage aux variables de chaînes, le TRI X/Y n'est plus aussi efficace:

```
3 HOME: HTAB(9): PRINT"APPLE TRUCS & ASTUCES"
4 PRINT: PRINT: HTAB(9): PRINT"R.PRUST / W. VOSS 1984": PRINT
10 REM *** ELABORATION D'UN REGISTRE ***
20 N=27
30 HOME: TEXT: DIM X$(N),Y(N)
40 FOR I=1 TO N: READ X$(I),Y(I): NEXT
50 GOSUB 1000
60 Z=1 : FOR M=Z TO N-1: IF X$(M)=X$(M+1) THEN NEXT M
70 IF M Z THEN A=Z : E=M : GOSUB 2000
80 PRINT X$(Z);: HTAB(20): PRINT Y(Z);: IF Z=M THEN 130
90 FOR Q=Z+1 TO M: IF Y(Q)=Y(Q)-1 THEN 120
100 PRINT",";: IF PEEK(36) 36 THEN PRINT: HTAB(20)
110 PRINT Y(Q);
120 NEXT Q
130 PRINT: IF M=N THEN END
140 IF M=N-1 THEN PRINT X$(N);: HTAB(20): PRINT Y(N): END
150 Z=M+1: NEXT M
160 END
1000 L=N-1
1010 L=INT((L-2)/3)+1: FOR  I=1  TO  N-L:  J=I+L:  HX$=X$(J):  HY=Y(J):  IF
X$(I) =HX$  THEN 1060
1020 Q=I
1030 X$(J)=X$(Q): Y(J)=Y(Q): J=Q: IF Q =L THEN 1050
1040 Q=Q-L: IF X$(Q) X$ THEN 1030
1050 X$(J)=HX$: Y(J)=HY
1060 NEXT I: IF L 1 THEN 1010
1070 RETURN
2000 H=14: Q=0
2010 IF E-A H THEN 2040
2020 FOR I=A TO E-1 : Y=Y(I): K=I: FOR J=I+1 TO E: IF Y(J) Y  THEN  Y=Y(J):
K=J
2030 NEXT: Y(K)=Y(I): Y(I)=Y: NEXT: GOTO 2140
2040 Y=A: J=E: HY=Y((I+J)/2)
```

```
2050 FOR I=I TO E: IF Y(I) HY THEN NEXT
2060 FOR J=J TO A STEP-1: IF Y(J) HY THEN NEXT
2070 IF I J THEN Y=Y(I): Y(I)=Y(J): Y(J)=Y: I=I+1: J=J-1: IF I J THEN 2050
2080 IF J-A E-1 THEN 2120
2090 IF I E THEN Q=Q+1: L(Q)=I: R(Q)=E
2100 E=J: IF A E THEN 2010
2110 GOTO 2140
2120 IF J A THEN Q=Q+1: L(Q)=A: R(Q)=J
2130 A=I : IF A E THEN 2010
2140 IF Q 0 THEN A=L(Q): E=R(Q): Q=Q-1: GOTO 2010
2150 RETURN
3000 DATA GOTO,13, REM,24,INDEX,234, GOTO,52,DIMENSION, 53,INPUT,12,GOTO,4,
    PRINT,12,PRINT ,78,INPUT ,90,GOTO,4
3010 DATA LEFT$, 198, MID$, 286, HTAB, 22, PRINT, 22, GOTO  ,66  ,VTAB,  2,
    IF..THEN,34, ARRAY, 92
3020 DATA ARRAY, 76, REM ,15, INPUT ,57, INDEX ,112, DIM, 160
3030 DATA GOTO ,234,GOTO ,143, VTAB, 28
```

Chapitre 8: Alternative pour la programmation en BASIC:

8.1 Remarques préliminaires:

Dans les chapitres précédents nous avons surtout traité des possibilités de l'APPLE II pour programmer en BASIC de façon précise, nous avons vu les problèmes de mise en oeuvre d'un ordinateur et maintenant on peut maitriser des problèmes plus compliqués. Le BASIC est d'un développement relativement récent et on ne peut pas lui demander la même capacité à résoudre les problèmes, qu'aux langages traditionnels.

On ne trouve pas pour tous les problèmes la solution soi-même, de nombreuses questions reviennent fréquemment, il faudrait les traiter plus longuement. C'est pourquoi il vaut mieux profiter des logiciels du commerce.

Les langages de programmation traditionnels et les logiciels sont deux des aspects que nous allons traiter dans ce chapitre. Il s'agit d'avoir une vue d'ensemble de ces problème plutôt que de mettre en place un programme particulier.

Il est possible que le lecteur de ce livre ne possède pas tous les éléments nécessaires pour résoudre toutes nos propositions. Mais nous offrons une vue d'ensemble qui permet d'évaluer les possibilités de développement d'un programme.

8.2 Autres langages de programmation

Lorsque l'on ne travaille plus avec l'interpréteur basic APPLESOFT et que l'on n'a pas chargé de programme en Basic, on peut utiliser d'autres langages de programmation. On se sert alors du programme de traduction correspondant (COMPILER) stocké sur la disquette. On dispose d'une capacité mémoire centrale suffisante (en générale au moins 48K).

Nous n'allons pas énumérer ici tous les différents langages de programmation. Nous ne traiterons que deux langages particuliers qui ont dans l'histoire de l'informatique, une place prédominante. Ce sont les langages

FORTRAN et COBOL

Ces deux langages existent déjà depuis des décennies et leur développement sert surtout pour des ordinateurs importants bien qu'on ait également élargi leur utilisation en effectuant une certaine modernisation, pour les petits ordinateurs.

FORTRAN est une abréviation pour FORMULA TRANSLATION (formule traduction). C'est un langage qui a été développé pour des techniques et des problèmes principalement mathématiques.

COBOL est une abréviation pour COMMON BUSINESS ORIENTED LANGAGE (langage orienté en direction du monde des affaires), il sert surtout à solutionner les problèmes EDV dans les domaines d'exploitation commerciale.

Nous allons simplement esquisser les propriétés et les qualités caractéristiques de ces langages pour donner au lecteur une idée des possibilités de ces langages. Nous ne prétendons en aucun cas présenter un cours complet sur le FORTRAN ou le COBOL, le lecteur s'étant peut-être

habituée au BASIC.

Nous commencerons par le FORTRAN qui, comme BASIC, est composé de différents dialectes. Nous traiterons également de sa nouvelle version, le FORTRAN 77.
C'est avec cette version que l'on a remplacé le FORTRAN IV utilisé depuis très longtemps mais qui n'était plus adapté à l'orientation moderne du dialogue de programmation.

On notera une certaine parenté entre les langages BASIC et FORTRAN.
Il y a ici aussi des groupes d'instructions définis: les instructions:

- pour l'entrée des informations .
- pour la sortie des résultats .
- pour les calculs .
- pour le traitement des textes.
- pour les branchements.
- pour l'édification de sous-programmes etc.....

En fait la différence avec BASIC réside surtout dans la forme: les instructions ne sont pas numérotées et quand on leur attribue un numéro il sert d'adresse de transfert. Une instruction FORTRAN est toujours introduite par 6 espaces si l'on a assez de place. Les numéros ne sont nécessaires que dans peu de cas, ils sont compris entre 1 et 5, 6 est prévu pour le marquage d'une ligne de reconduction si l'instruction est trop longue.

Maintenant nous allons examiner les instructions importantes et leur construction formelle. Nous placerons à côté celles de BASIC correspondantes, ce qui permettra au lecteur de comprendre plus facilement. Cette énumération n'est pas complète.

Les instructions de FORTRAN les plus importants sont les suivants:

1. Entrée des données:

READ *, liste de variables

On entre un petit nombre de données.

READ (n,m) liste de variables

Des données plus importantes (et formatées) sont lues. N contient le numéro de la donnée en entrée, et M l'indication de format qui lui correspond.

2- Formatage:

m FORMAT (Spécifications)

Avec cette instruction les données sont décrites et lues.
Après le descriptif de toutes les instructions, nous fournirons des explications plus détaillées sur celle-ci.
FORMAT doit être introduite avec un numéro qui correspond à l'instruction READ.

3- Sortie de données:

PRINT *, liste de variables

Instruction pour la sortie d'un nombre réduit de résultats.

WRITE (n,m) liste de variables.

Sortie d'une liste de résultats formatée plus importante.

4- Instructions arithmétiques:

Variable= Expression arithmétique

C'est une instruction simple, elle correspond au LET du BASIC.

5- Branchement conditionnel:

IF (expression logique) Suite d'instructions

Cette instruction correspond au IF...THEN du BASIC, on peut également utiliser d'autres opérateurs logiques:

.EQ. correspond à =
.NE. "
.LE. " =
.GE. " =
.LT. "
.GT. "

6- Branchement inconditionnel:

GO TO m

Cette instruction est semblable au GOTO du BASIC.

7- Branchement arithmétique:

IF (expression arithmétique) n1, n2, n3

Cette instruction arithmétique IF aiguille à la ligne n1, si l'expression arithmétique est négative , à la ligne n2 si elle est égale à 0, à la ligne 3 si elle est positive.

8- Boucle:

DO n, Variable=Début, Fin

Une boucle est introduite avec cette instruction (correspond au FOR...TO du BASIC),

n CONTINUE

Instruction pour répétition de la boucle (correspond au NEXT du BASIC)
n doit correspondre au n du DO.

9- Dimensionnement:

DIMENSION Nom de la variable (Valeur)

correspond au DIM BASIC.

10- Préparation des données:

DATA Liste de variables/Valeur/

permet le pointage des données sans pour cela être lues par READ. Si l'on voulait le comparer au BASIC, il faudrait coupler les instructions DATA et READ.

CHARACTER variable = impression

Avec cette instruction en FORTRAN, on peut pointer les textes (ce qui correspond au BASIC: LET Nom variable$ = chaîne).

11- Sous-programme:

SUBROUTINE Nom (Argument)

C'est avec cette instruction que l'on introduit des sous-programmes (qui de même que pour le programme principal, se terminent avec END).

L'appel du sous-programme se fait avec :

CALL NAME (Argument)

Ceci correspond au GOSUB en BASIC.

12- Fin de programme:

Il se termine avec

END

comme en BASIC.

13- commentaires:

Cette énumération très incomplète s'achève avec l'équivalent du REM BASIC. Tous les textes qui sont introduits avec un C en première position, servent de commentaire et se traitent comme REM en BASIC.

En FORTRAN à la différence du BASIC on utilise une instruction FORMAT. Elle sert à la lecture d'une ligne de données et en partie aux sorties à formater.
Pour mieux comprendre ce procédé, voici 2 lignes de données:

```
Nr.  Nom         Age     Taille      Poids
-----------------------------------------
17   DUPONT      42       1.85        84.5
18   DURAND      38       1.70        75
-----------------------------------------
```

En BASIC ces lignes de données seraient transmises à l'ordinateur de la manière suivante:

```
10 DATA 17,DUPONT,42,1.85,84.5
20 DATA 18,DURAND,,38,1.70,75
...
499 Z=2
500 FOR I=1 TO Z
510 READ NR(I),N$(I),A(I),GR(I),GW(I)
420 NEXT I
```

En FORTRAN, on garde en général l'énoncé des données et en "fixed format" c'est à dire avec des longueurs de champs constantes:

17 DUPONT 421.8584.5

18 DURAND 381.7 75

Ceci implique que l'on communique à l'ordinateur les divisions du champ dans le programme, et aussi le type de variables du champ (réelle numérique, entière ou alphanumérique).
C'est ce qui se passe avec FORMAT.

En FORTRAN on suit une règle générale pour la mise au point:
Tous les noms des variables qui commencent avec les lettres I à N, sont traités comme des variables entières (I-N=IN integer=entier); toutes les autres comme des variables réelles (si on a besoin de dévier de ceci il existe une instruction mais nous n'en parlerons pas ici).

On traite les variables entières avec le format I, les variables réelles avec le format F, les variables de chaînes avec A (il y a de nombreuses autres possibilités: pour la transmission de champs ou pour les changements de lignes, nous ne nous étendrons pas sur ces maniements).

Les lignes de données ci-dessus peuvent être décrites avec une seule des instructions format suivante (à cause de "fixed format"):

Format (I2,A15,I2,F4.2,F4.1)

ceci signifie :

I2 : Le premier champ occupe 2 places et il s'agit d'un nombre entier (I pour "integer").

A15 : Le deuxième champ est un champ alphanumérique pour lequel en général 15 places sont réservées.

I2 : Le troisième champ est de nouveau un entier de 2 emplacements.

F4.2 : On attend un réel occupant 4 places (avec le point décimal) dans ce champ. Les 2 emplacements derrière sont des emplacements décimaux.

F4.1 : Idem mais il n'y a qu'un emplacement décimal après le point.

On lit de la manière suivante :

```
      7
C     LECTURE
      Z=2
      DO 1 I=1,Z
      READ (5,99)NR(I), N(I), A(I), GR(I), GW(I)
  99  FORMAT (I2,A15,I2,F4.2,F4.1)
 1    CONTINUE
C     SUITE DU TRAITEMENT
      ...
```

Dans cette partie de programme il suffirait de changer Z si l'on voulait lire un nombre de données plus important. Ensuite on dimensionne comme c'est le cas en basic .

Pour finir voici un exemple en FORTRAN avec sa correspondance en BASIC:

```
C    PROGRAMME POUR CALCUL DE LA SOMME
C    DES NOMBRES ENTIERS DE 1 A 100
C
C
      DO 10 I=1,100
      S=S+1
   10 CONTINUE
      WRITE (6,100)5X,"SOMME =",S
  100 FORMAT 16
      END
```

En BASIC:

```
10 REM PROGRAMME POUR CALCUL DE LA SOMME
20 REM DES NOMBRES ENTIERS DE 1 A 100
```

```
30 REM
40 FOR I=1 TO 100
50 S=S+1
60 NEXT I
70 PRINT TAB(5) "SOMME=",S
80 END
```

Nous allons maintenant examiner le COBOL, il montre certaines ressemblances avec le BASIC mais moins que le FORTRAN. Nous n'allons énumérer ces instructions que rapidement mais ceci permettra de voir le déroulement d'un programme.

La première différence avec le BASIC et le FORTRAN vient de l'édition. Chaque programme COBOL est divisé en 4 parties que l'on appelle DIVISION:

1-IDENTIFICATION DIVISION
2-ENVIRONMENT DIVISION
3-DATA DIVISION
4-PROCEDURE DIVISION

Ces 4 divisions sont introduites par leurs titres. Elles sont elles-même divisées en paragraphes. Les noms de ces paragraphes servent d'adresse de transfert. Les parties de programme qui doivent être transférées pendant le traitement du programme doivent de leur côté disposer de marques en tant que paragraphes,pour pouvoir disposer d'une adresse de transfert.

D'un point de vue formel, l'instruction COBOL doit commencer en position 8 ou 12, colonne A et colonne B. Chaque instruction doit être terminée par un point.

Que se passe-t'il dans ces divisions ?

Il faut d'abord s'assurer qu'il y a des instructions dans chaque division

car elles sont obligatoires et doivent être données par le programmateur. D'autres instructions ne seront relevées qu'en cas de nécessité.

Nous allons nous occuper plus particulièrement de l'IDENTIFICATION DIVISION.

Le programme porte un nom, il est possible d'utiliser d'autres instructions qui permettent une meilleure compréhension du programme.
L'affectation d'un nom se fait avec l'instruction :

PROGRAM-ID. Nom (un commentaire quelconque)

Toutes les autres entrées sont au choix. Les énoncés dans l'IDENTIFICATION DIVISION commencent colonne A, soit en position 8 et doivent être fermés par un point et un espace. (Une éventuelle ligne de reconduction commence en position 12 ou colonne B.)

Un programme COBOL peut commencer de la manière suivante :

```
8
IDENTIFICATION DIVISION.
PROGRAM-ID. PROGRAMME No.1.
```

Dans la division ENVIRONMENT, la configuration utilisée est décrite, c'est-à-dire que les unités INPUT, OUTPUT,et le cas échéant les dispositifs de mémoire sont dénommés avec leur numéro d'unité logique.

On définit en particulier dans cette division sur quel système le programme est compilé et sur lequel il est exécuté. C'est pourquoi la relation entre les fichiers utilisés (souvent INPUT et OUTPUT) et les unités périphériques doivent être établie pour que le fichier soit compétent.

La relation et les numéros des unités périphériques peuvent être établis

différemment. Cela dépend du contenu de l'ENVIRONMENT DIVISION utilisé. Cette division est sous-divisée en 2 sections:

1- CONFIGURATION SECTION
   (elle décrit les installations utilisées)
2- INPUT-OUTPUT SECTION
   (décrit l'assignation du dispositif de fichier)

Le programme commencé plus haut pourrait être poursuivi ainsi:

8

```
(1)     IDENTIFICATION DIVISION.
(2)     PROGRAM-ID. PROGRAMME No.1.
(3)     ENVIRONMENT DIVISION.
(4)     CONFIGURATION SECTION.
(5)     SOURCE COMPUTER. APPLE.
(6)     OBJECT COMPUTER. APPLE.
(7)     INPUT-OUTPUT SECTION.
(8)     FILE-CONTROL.
(9)         SELECT ENTREE ASSIGN TO DISPOSITIF.
(10)        SELECT SORTIE ASSIGN TO DISPOSITIF.
```

Pour pouvoir expliquer le découpage du programme, la partie gauche a été numéroté:

```
-----------------------------------------
I CETTE NUMEROTATION N'EST PAS UN ELEMENT I
I     CONSTITUANT DU PROGRAMME COBOL.     I
-----------------------------------------
```

Les lignes (1) et (2) sont déjà connues.
La deuxième division est introduite avec la ligne (3); la première section

par la ligne (4).

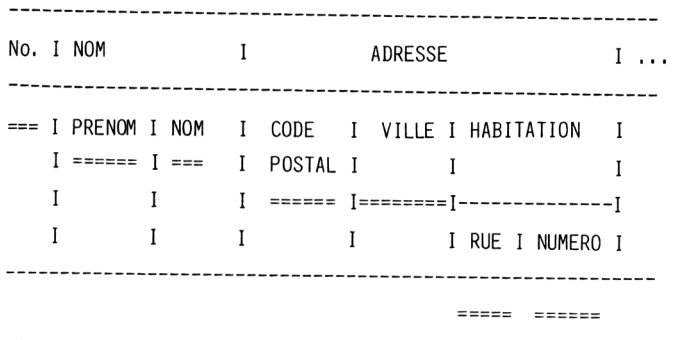
En ligne (5) est défini l'ordinateur de traduction, en ligne (6) celui qui doit calculer les données. Les deux doivent être identiques. L'instruction de cette séquence ne peut pas être transformée.
Le signe de reconnaissance de la deuxième section de la deuxième division apparaît dans la ligne (7). La première instruction obligatoire est: FILE CONTROL, qui insère un fichier d'assignation.

En ligne (9) comme en (10) qui est quasiment une sous-partie de l'instruction FILE CONTROL, les dispositifs sont déclarés en position 12 de la manière suivante:

SELECT Nom du fichier ASSIGN TO numéro de dispositif.

Quel que soit le cas, il faut consulter le manuel d'utilisation.

Dans la DATA DIVISION, la construction d'une ligne de données est définie (c'est comparable au FORMAT du FORTRAN). Le concept de niveau joue un rôle important: chaque ligne de données est sous-divisée en groupe de données, qui de nouveau l'est en sous-groupe, etc...
Nous allons illustrer ceci avec un exemple de fichier de clients:



Il s'agit d'une sous-division à 3 paliers et celle-ci n'est pas conservée pour chaque groupe de données. De même que dans l'instruction FORMAT du FORTRAN, l'ordinateur procède au partage des champs avec PICTURE CLAUSE (PICTURE = IMAGE). Voici un exemple tirée des données ci-dessus:
Exemple de concept de niveaux:

```
01 LIGNE DE CLIENTS.
   02 NR PICTURE IS 9(4).
   02 NAME.
      03 AVANT PICTURE IS X(10).
      03 APRES PICTURE IS X(15).
   02 ADRESSE.
      03 CODE POSTAL PICTURE IS 9(4).
      03 VILLE PICTURE IS X(20).
      03 HABITATION.
         04 RUE PICTURE IS X(15).
         04 4 NUMERO PICTURE IS 9(3).
```

Avec cette construction schématique d'une ligne de données, on a obtenu une sous-division telle qu'elle a été établie plus haut.

Ces champs qui sont encore sous-divisés ne contiennent pas de PICTURE (exemple 02 NAME). Ceux qui ne sont pas sous-divisés (par exemple 03 code

postal) contiennent une PICTURE qui indique à l'ordinateur comment se présente le champ (ces champs ont été soulignés dans le schéma ci-dessus). Le chiffre 9 indique un champ numérique, la lettre X est utilisée pour un champ alphanumérique (il y a de nombreuses autres possibilités, par exemple: marquer un point décimal et un emplacement décimal etc....).

PICTURE IS 9(4) signifie que nous allons examiner un champ numérique à 4 emplacements (à la place de 9(4) on aurait pu écrire 9999).

Dans les bords d'impression on ne se sert pas du concept de niveau car on ne doit entrer ou sortir que des petites quantités de données (exemple dialogue orienté), nous reparlerons de ce sujet plus tard.

Pour l'entrée des données cette structure est efficace et il doit en découler le résultat du processus de traitement des données. Si pour ce processus on utilise plus de 2 fichiers (INPUT et OUTPUT) il faut établir d'autres structures. Ces structures doivent être produites dans la première section de la DATA DIVISION qui porte le nom de FILE SECTION. Sa première instruction est FD (File Declaration) :

FD Nom de fichier, Description du fichier.

Dans la description du fichier on trouve le nombre de lignes de données, leur longueur, si cette ligne de données est séquentielle ou non-mémorisée, et comment le fichier est protégé. Nous ne parlerons pas des particularités.

La deuxième section de DATA DIVISION est le WORKING STORAGE SECTION: les emplacements de mémoire pour toutes les données sont réservés et décrits, les entrées-sorties du fichier ne sont pas gérées. C'est un champ d'aide qui sera utilisé après l'entrée ou avant la sortie du processus de traitement des données: par exemple pour l'entrée-sortie d'un champ contenant des petites quantités de données qui ne sont pas organisées en

fichier.

Dans la section WORKING STORAGE, on travaille avec le concept de niveaux, l'instruction FD disparaît.

Les valeurs isolées qui ne sont pas à classer dans le concept de niveaux, ont pour numéro: 77 (il peut être utilisé dans FILE SECTION).

Exemple:

```
--------
       8
       WORKING-STORAGE SECTION
       77 DONNEE1 PICTURE IS 9999.
       77 DONNEE2 PICTURE IS 9999.
       01 LIGNE.
          02 MULT PICTURE IS 9(8).
          02 DIV  PICTURE IS 999V99.
```

(le V dans la dernière PICTURE est la marque de l'emplacement d'un point décimal).

Maintenant voyons la dernière division: PROCEDURE DIVISION. C'est là que se trouve les instructions pour le travail des entrées:

Voici les plus importantes:

1-Ouverture d'un fichier:

```
-------------------------
```

Les entrées-sorties d'un fichier sont ouvertes avec l'instruction OPEN.

```
Exemple:     OPEN INPUT ENTREE.
             OPEN OUTPUT SORTIE.
```

ou: INPUT
OPEN OUTPUT Nom de fichier.

2-Fermeture d'un fichier:

Tout fichier ouvert doit être refermé à la fin du programme par:

CLOSE Nom de fichier.

3-Lecture d'un fichier en entrée:

Les lignes de données d'un fichier en entrée sont lues avec l'instruction READ:

READ Nom de fichier RECORD AT END Instruction.

4-Sortie de lignes de données:

Les résultats sont écrites avec l'instruction WRITE:

WRITE Nom de la ligne de données.

A l'inverse de l'instruction READ on ne sert pas d'un nom du fichier mais du nom de la ligne de données (elle comprend la marque du champ qui porte dans le concept de niveaux, la description de sortie du fichier dans la division DATA: le numéro de niveau 01).
En fait la différence est que par READ le fichier en entrée est lu (et doit l'être) alors que par WRITE chaque ligne de données est écrite (toujours une par une).

5- L'instruction MOVE:

MOVE champ d'émission TO champ de réception.

correspond au LET du BASIC. Il existe une possibilité pour l'assignation des valeurs concernant des petites quantités de données dans la section WORKING STORAGE de la DATA DIVISION.

Exemple:

```
--------
       WORKING-STORAGE SECTION
       77 DONNEE1 PICTURE IS 9999 VALUE IS 1234.
       77 NOM1    PICTURE IS X(4) VALUE IS 'VOSS'.
```

6- Expressions arithmétiques:

```
-----------------------------
```

Les 4 opérations seront effectuées en COBOL, d'après le modèle suivant:

```
Addition:       ADD a TO b.
                Opération: b=a+b

Soustraction:   SUBTRACT a FROM b.
                Opération: b=b-a

Multiplication: MULTIPLY a BY b.
                Opération: b=a*b

Division:       DIVIDE a INTO b.
                Opération: b=b/a
```

Il existe une autre possibilité de formulation, valable pour ces 4 opérations:

```
                ADD a b GIVING c
                Opération: c=a+b
```

On dispose de l'instruction COMPUTE pour les opérations compliquées:

COMPUTE a=expression arithmétique.

7- Branchement conditionnel:

GOTO Liste DEPENDING ON Variable.

Si Variable a la valeur 1, on se branche sur le premier nom de la liste;
Si Variable a la valeur 2, dans le deuxième nom de liste etc...

L'instruction IF correspond à:

IF Condition Instruction
NEXT SENTENCE (ELSE...)

Cette instruction fonctionne sur le même principe que le IF...THEN (ELSE...) du BASIC. On peut utiliser comme opérateur de comparaison: = .

8- Fin du programme:

On achève un programme COBOL avec l'instruction

STOP RUN.

Ce sont les plus importantes instructions COBOL.

Pour compléter ces instructions voici un exemple pour les petites quantités de données. Dans la DATA DIVISION il faut atteindre l'emplacement section WORKING STORAGE et constituer un concept de niveau compliqué devient superflu:

Entrée : ACCEPT Nom de variable.
Sortie : DISPLAY ('Texte') Nom de variable.

Voici un exemple simple et le programme BASIC correspondant:

```
8
IDENTIFICATION DIVISION
PROGRAM-ID.ELEVATION AU CARRE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE COMPUTER. APPLE.

OBJECT COMPUTER. APPLE.

DATA DIVISION.

WORKING STORAGE SECTION.
77 ENTREE PICTURE IS 9(4).
77 SORTIE PICTURE IS 9(8).

PROCEDURE DIVISION.
DEBUT.
    ACCEPT ENTREE.
    COMPUTE SORTIE=ENTREE+ENTREE.
    DISPLAY 'CARRE=',SORTIE.
FIN.
    STOP RUN
```

Un seul nombre est élevé au carré et le résultat est sorti directement. En BASIC, voici le programme équivalent:

```
10 INPUT "SVP UN NOMBRE : ";E
20 A=E*E
30 PRINT "CARRE=";A
40 END
```

Ce programme BASIC apparaît en comparaison beaucoup plus simple. Les avantages de la programmation en COBOL n'apparaissent en fait que lorsqu'il y a des quantités très importantes de données à traiter, de plus il faut connaître les possibilités particulières et supplémentaires du COBOL.

8.3 Les logiciels

Pour un micro-ordinateur comme l'APPLE II il existe de nombreux logiciels de qualité. Ces logiciels sont tellement nombreux qu'il sera tout à fait impossible d'en donner la liste exhaustive ici.
On peut classer les logiciels en fonction des différents domaines pratiques auxquels ils s'adressent et, d'autre part, en fonction des activités principales sur lesquelles ils s'appuient.
Voici les différents domaines d'applications:(là encore, on ne prétendra pas donner une liste complète):

1. Programmes de résolution de problèmes mathématiques.
2. Programmes de gestion de données (Fichiers clients, gestion de stocks...)
3. Programmes statistiques
4. Programmes d'assistance pour le bureau et l'administration.
5. Programmes pour l'établissement de projets, plannifications, esquisses, dessins...
6. Programmes pour la commande de robots et automates.
7. Programmes de jeux (à ne pas oublier, même si moins important)

Puis les différentes activités:

1. Programmes de calcul.
2. Programmes de traitements de textes.
3. Programmes graphiques.

Voici un exemple:
Les représentants les plus importants des programmes de calcul sont:

Visicalc (prévu pour APPLE II)
et MULTIPLAN

MULTIPLAN est un programme de calcul pour APPLE II, il sert à évaluer des

tableaux , planifier des budgets, calculer des tendances etc... NOus n'entrerons pas dans le détail).

Il est possible de mettre en route l'APPLE II avec un autre système d'exploitation grâce à une carte optionnelle, on peut ainsi utiliser des programmes de calcul très performants (programmes pour architectes, médecins, éditeurs, assureurs...). Nous n'en parlerons pas plus.

Certains programmes sont développés pour des problèmes mathématiques spécifiques: calcul de vecteurs et de matrices, calcul différentiel et intégral...

Les statistiques représente un secteur particulier de ce domaine.

Voici les programmes de ce secteur à notre disposition:

1. Dialogue orienté, entrée de données, contrôle de données et analyse statistique descriptive (ex. Système DATAN II de R. Schnell)

2. Statistiques interférentes, analyse de données par la mise en oeuvre d'un procédé de test statistique (ex. Système REPTEST de R. Prust)

3. Procédés d'analyse statistique multi-variables (ex. Système TAXONOM de Weber)

etc...

On dispose également de programmes de traitement de textes compatibles avec le système d'exploitation de l'APPLE soit demandant un autre système d'exploitation.

Les programmes APPLE WRITER et SCRIPT font partie du premier groupe, WORDSTAR appartient au second groupe.

Avec les traitements de textes on peut rédiger des lettres, des textes, des opérations...Les modifications d'écriture, les corrections d'erreurs, adjonctions ou suppression de textes, notes en bas de page,etc...sont possibles sans problème.On notera les particularités dans le manuel livré

avec le logiciel.

Un groupe de logiciels sert à la gestion de fichiers. C'est-à-dire tous les programmes qui permettent la gestion de données avec lesquelles les fichiers sont établis.
Il permet l'extension, la réduction, la modification du fichier ainsi que la recherche de données grâce à des critères de recherche et des des mots clés.
Les programmes QUICKFILE et PROFIT II en sont des représentants typiques.

Le dernier groupe est celui des programmes graphiques.
Comme on l'a déjà vu au chapitre 2, l'APPLE dispose de possibilités graphiques et d'une série d'instruction permettant l'édition de graphiques.
Il existe de plus des programmes d'aides au graphisme utiles (Courbes, graphiques à barres, en camembert, graphiques de couleur, hachurés...) et d'autres permettant la construction de dessins d'architecture et artistiques (Nous ne parlons pas des dessins artistiques employés dans de nombreux jeux).
Ces programmes tels que Business Graphics ou CHARTMASTER (pour APPLE II et III) appartiennent au premier groupe.
Appartiennent au second groupe, des programmes qui apparaissent sur le marché et qui ont été développé pour la programmation CAD (CAD=Computer Aided Design= Développement et construction assistée par l'ordinateur).
Nous n'insisterons pas sur le sujet.

ANNEXES

Annexe I: Messages d'erreur

BAD SUBSCRIPT ERROR

Une variable utilise un indice supérieur à ce qu'il a été prévu dans DIM ou le nombre d'indices est différent de celui spécifié dans DIM.

CAN'T CONTINUE ERROR

La commande CONT ne peut être exécutée parce que des modifications du programme ont eu lieu.

DISK FULL

La disquette est pleine. On ne peut plus mémoriser de données supplémentaires.

DIVISION BY ZERO ERROR

Tentative de division par zéro.

END OF DATA

Tentative de lecture d'une donnée qui n'est pas présente sur la disquette.

FILE LOCKED

Tentative de modification de la protection en écriture du fichier.

FILE NOT FOUND

Fichier inexistant.

FILE TYPE MISMATCH

On s'adresse à un fichier avec une instruction inappropriée.

ILLEGAL QUANTITY ERROR

Valeur en dehors de l'intervalle permis.

I/O ERROR

Une disquette est adressée mais pas atteinte (ex. une disquette non initialisée)

NEXT WITHOUT FOR

Une instruction NEXT est trouvée sans l'instruction FOR...TO correspondante.

OUT OF DATA ERROR

On cherche à lire plus de données qu'il n'y en a.

OUT OF MEMORY ERROR

Capacité mémoire insuffisante pour un programme défini.

OVERFLOW ERROR

Un nombre est supérieur à 1.7E+38 .

REDIM'D ARRAY ERROR

On cherche à redimensionner une variable.

RETURN WITHOUT GOSUB ERROR

Une instruction RETURN est trouvée sans instruction GOSUB.

SYNTAX ERROR

Erreur de formulation dans une instruction ou erreur sans message d'erreur spécifique.

TYPE MISMATCH ERROR

Une valeur alphanumérique a été affectée à une variable numérique ou le contraire.

UNDEFINED FUNCTION ERROR

On a essayé d'appeler une fonction non définie.

UNDEFINED STATEMENT ERROR

Branchement sur une ligne inexistante.

Annexe II: Sous-programmes importants

CALL - 1052	Emission d'un BIP
CALL - 1008	Dépacement du curseur vers la gauche
CALL - 998	Déplacement du curseur sur la ligne précédente
CALL - 922	Déplacement du curseur sur la ligne suivante
CALL - 868	Effacement des caractères à droite du curseur
CALL - 336	Entrée de l'interpréteur BASI
CALL - 211	Affiche ERR et émet un BIP
CALL - 151	Entrée du moniteur

Annexe III: Adresses mémoires

Adresses		Fonction
Décimal	Hexadécimal	
32	$20	Colonne gauche de la fenêtre écran Valeurs autorisées: 0...39
33	$21	Largeur de la fenêtre Valeur autorisées:0...(40-PEEK(32))
34	$22	Ligne la plus haute de la fenêtre Valeurs autorisées: 0...23
35	$23	Dernière ligne de la fenêtre Valeurs autorisées: 0...23 avec PEEK(34) PEEK(35)
36	$24	Position horizontale du curseur Valeurs autorisées: 0...PEEK(33)
37	$25	Position verticale du curseur Valeurs autorisées: 0...23
50	$32	Mode de l'écran: Valeurs autorisées: 63 - INVERSE 127 - FLASH

255 - NORMAL

1024	$400	Page 1 de basse résolution
à	à	
2047	$7FF	Valeurs admises:tous les caractères en code écran ASCII.

Annexe III

Adresses		Fonctions
Décimal	Hexadécimal	
-16368 (49168)	$C010	Drapeau d'entrée manuelle
-16384 (49152)	$C000	Code du dernier caractère entré
103/104	$67/68	Début du programme
105/106	$69/6A	LOMEM - Plus petite adresse Basic utilisable
115/116	$73/74	HIMEM - Plus haute adresse Basic utilisable
175/176	$AF/B0	Première adresse après la fin de programme / Début des variables
109/110	$6D/6E	Fin de la zone des variables numériques
218/219	$DA/DB	Numéro de ligne correspondant à la

dernière erreur

222	$DE	Code de la dernière erreur
216	$D8	Drapeau pour ONERR...GOTO

Annexe IV: Organisation de la mémoire de l'APPLE II

Adresses		Type	Utilisation
Décimal	Hexadécimal	Mémoire	
0-255	0-FF	RAM	Programme système
256-511	100-1FF	RAM	Pile
512-767	200-2FF	RAM	Tampon clavier
768-1023	300-3FF	RAM	Vecteurs du moniteur
1024-2047	400-7FF	RAM	Données pour 1ère page texte et graphisme basse résolution
2048-3071	800-BFF	RAM	Données pour 2ème page texte et graphisme basse résolution
3072-8191	C00-1FFF	RAM	Libre
8192-16383	2000-3FFF	RAM	Données pour 1ère page de graphisme haute résolution
16384-24575	4000-5FFF	RAM	Données pour 2ème page de graphisme haute résolution
24576-49151	6000-BFFF	RAM	Libre

49152-53247	C000-CFFF	RAM	Entrées/Sorties
53248-65535	D000-FFFF	ROM	Interpréteur BASIC-Integer Interpréteur APPLESOFT Moniteur...

Achevé d'imprimer en juillet 1985
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : juillet 1985
No d'imprimeur : 507030

IMPRIME EN FRANCE

Très important : l'ouvrage APPLE 2 TRUCS ET ASTUCES repose sur l'expérience acquise dans le travail avec le 2+, 2E et le nouveau super compact 2C. Les PEEKS et POKES intéressants, les bases de la programmation en assembleur, le graphisme couleur, la structure des masques-écran ne sont que quelques-uns des nombreux thèmes abordés. Un aperçu sur l'utilisation de logiciels écrits pour l'APPLE 2 complète ce nouvel ouvrage que tout possesseur d'APPLE 2 doit se procurer.

ISBN : 2-86899-021-5

Prix : 149 F TTC
Ref. : ML 111

APPLE 2
TRUCS ET ASTUCES